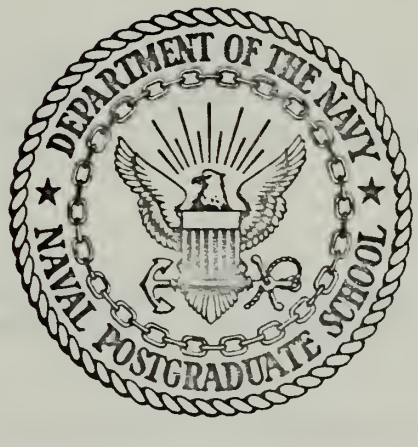
GPGL: A MODEL INTERACTIVE, GENERAL PURPOSE GRAPHIC LANGUAGE

James Dale Beans

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

GPGL: A MODEL INTERACTIVE, GENERAL PURPOSE GRAPHIC LANGUAGE

by

James Dale Beans

Thesis Advisor: R.D. DeLaura

December 1971

GPGL: A Model Interactive, General Purpose Graphic Language

by

James Dale Beans
Major, United States Marine Corps
B.S., United States Naval Academy, 1957

Submitted in partial fulfullment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1971

ABSTRACT

General Purpose Graphic Language (GPGL) is an interactive language which is intended for both two-dimensional and three-dimensional displays. The thesis contains a survey of the attributes and capabilities of an interactive general purpose graphic language. The more popular general purpose graphic languages are compared and the results included. The system and user-defined functions (including the construction of user-defined functions) of GPGL are explained. The implementation of a subset of GPGL at the Naval Postgraduate School on an Adage AGT-10 graphics terminal is described. The main purpose of implementing a selected subset of functions from GPGL is to examine the tri-level hierarchy established within the components of the graphical display; the manner in which this hierarchy is implemented is addressed in the thesis.

TABLE OF CONTENTS

ACKNOWLEDGMENT

The author wishes to express his appreciation to William Thomas of the Electrical Engineering Computer Laboratory, Naval Postgraduate School for his time and patience in assisting in the utilization of the Adage Graphics Terminal.

I. INTRODUCTION

With the advent of the digital computer and the expansion of the multitude of applications for the computer, the field of "Computer Graphics" has become of prime importance. Computer graphics refers to the use of a display device (usually a cathode ray tube) with auxiliary devices connected on-line to a computer. The cathode ray tube or other display device is used for graphic communications with the computer [1]. Computer graphics really came of age in 1963 when Sutherland used his Sketchpad graphic system to demonstrate the designing of various linkages and the analysis of the structural stress in a bridge.

As the years have passed and the uses for computer graphics have increased at almost an exponential rate, the need for a graphic language or languages has increased proportionally, therefore, the number of graphic languages since Sutherland's demonstration of the feasibility of computer graphics has greatly increased in the last eight years. These graphic languages range from extensions of common high-level programming languages (e.g., FORTRAN, ALGOL, etc.) to many highly specialized graphic languages which are designed to be used in only one area of application. Naturally, interest has been generated in general purpose graphic languages which could be used to assist in the many applications which require or use computer graphics.

The purpose of this thesis is threefold:

(1) To discuss certain considerations taken into account in selecting and designing the model language;

(2) To present General Purpose Graphic Language (GPGL);

(3) To discuss the implementation of General Purpose Graphic System (GPGSY), a basic subset of GPGL.

It was determined that GPGL would be, as much as possible, hardware independent. The only real hardware requirements in addition to a digital computer is that the hardware included a display device (general requirement for computer graphics), which is normally a cathode ray tube, some type of input/output attention device, (e.g., a light pen, mouse, joy stick etc.,)and a teletypewriter. Some type of input/output attention device and teletypewriter are normal components of a computer graphics system. It is intended that the GPGL be able to be implemented in its entirety (which will not be feasible in many cases because of the anticipated large core memory and/or auxiliary memory requirements) or partially implemented by selecting a desired subset as was accomplished at the Naval Postgraduate School. GPGL was designed to be extendable, meaning the user can develop more complex functions if he desires in accordance with his own programming skills. Lastly, GPGL is designed so that it can be utilized by students who have little or no programming experience or knowledge.

The thesis is devided into three parts. First, this thesis deals with the considerations taken into account in

determining GPGL. This includes a brief look at the development of graphic languages, what a graphic language is, and the more important attributes required of a general purpose graphic language. The latter includes a selection of the graphic capabilities which are required in a graphic language. The second part covers the functions of GPGL with the necessary description of what the specific functions accomplish, the inputs required for the function, and examples demonstrating how the language could be used. The last section deals with the implementation of GPGSY, an actual subset selected from GPGL, the directions for the use of the GPGSY, some examples of its use, and some of the more important matters considered in the implementation phase. The Computer Program used to accomplish the implementation is appended for further reference.

II. CONSIDERATIONS IN DESIGNING THE LANGUAGE

A. THE DEVELOPMENT OF GRAPHIC LANGUAGES

Graphic languages have not developed as rapidly as the more familiar programming languages. Probably the first uses of graphics were made in the early 1950's with the Whirlwind computer. In 1955, the APT (Automatically Programmed Tools) language was demonstrated on the Whirlwind. Even though APT is a specialized programming language, it does have the ability to be used in conjunction with computer graphics [2]. After 1957 when FORTRAN became popular as a normal programming language, and as computer graphics grew, it was natural for FORTRAN to be extended for computer graphics. This was done through the language GRAF [3]. As ALGOL became popular it was also extended for computer graphics under the name AED (ALGOL Extended For Design) [4]. Still in the development states at The Rand Corporation is an extension to a conversational subset of PL/I which could be used for computer graphics. The latter would give the user the ability to program in a conversational mode in the more powerful PL/I language [5].

An even more popular approach was the development of system graphic subroutines. At first these were designed to be used exclusively with FORTRAN. Examples of these are GSP [6] and DISPLAYTRAN [7]. IGS (Integrated Graphics System) was developed by The Rand Corporation and is a

subroutine package which the user can use with FORTRAN, PL/I or any of the other languages which have standard IBM Operating System/360 linkages [8].

Many other graphic languages were developed independently from the normal high-level programming languages. The early computer graphics system took the direct approach to "syntactic" representation, that is the display itself constituted sufficient representation [9]. As such, the dynamic graphical languages of SKETCHPAD [10] and CADET (Computer Aided Design Experimental Translator) [11] had a syntax of FUNCTION, BUTTON1, FUNCTION, BUTTON2 where FUNCTION was the selection of a function and BUTTON was the designation of the parameters of the function. This type of description was hard to explain and understand and is obviously very hardware-oriented. CADET is of interest because it demonstrated for the first time that a dynamic graphical language could be handled in the same manner as a verbal programming language. By developing a data structure of the binary tree type and by using a precedence table with many precedence pointers, the originators of CADET illustrated they could display a picture from data structure information. They showed that the process of constructing a display list to generate a display on a CRT from a data structure is analogous to the generation by a compiler of specific machine code instructions from source code statements [11]. More recently, graphic languages have also used metacompilers, compilers, interpreters, and

subroutine calls. The question of which to use will be examined in more detail later.

Many specialized graphic languages have been developed to handle specific application areas. CAFE is a language which was specifically designed to be used in the making of motion pictures and uses SNOBOL to handle the conversational mode between the user and the computer and to construct the data structure. FORTRAN is used to process the final data structure and perform the perspective transformations [12]. BUGSYS [13] and PDL [14] are languages or systems specifically designed to analyze and process pictures. These are just two of many specialized languages.

Only a few general purpose languages have been developed independently from the high-level programming languages. One of these languages was developed by Kulsrud [15]. Kulsrud's language is not only designed to construct displays, but is specifically designed to handle both topological and pictorial analysis. Kulsrud's general purpose language with its metacompiler is a good representative language of the present state of the art.

B. DEFINITION OF A GRAPHIC LANGUAGE

In the preceding paragraphs numerous references have been made to graphic languages which actually refer to graphic systems in toto (e.g., Sketchpad, IGS). In another case a translator, CADET, was referred to as a language. This ambigious definition of a graphic language is common

throughout the field of computer graphics. Morrison states:

> "The term "graphic language" has been used ambiguously, in the literature, to describe at least three different types of language used in graphic processing.
>
> 1. The input stream is in the form of actions taken by a console operator.
>   a. draw with light pen
>   b. type names and numbers
>   c. push buttons
>   d. light pen references of objects on the screen
>
> A language translator translates these actions into invocations of appropriate procedures. These procedures perform requested actions and provide displayed feed back to the user.
>
> 2. Input is in the form of pictures existing on film or other media. In this case, the language translator is a pattern recognizer which recognizes and extracts meaning from these pictures.
> 3. A set of programming tools (functions and subroutines) are embedded in a "host" language (e.g. FORTRAN). Using these tools lightens the load of the graphic system." [16]

When complete graphic systems are referred to, more than just the graphic language is included. Sketchpad, which is a program written for the TX-2 computer, is a complete system, not just a graphic language. It includes a complicated, ring-type data structure. The many different types and forms of data structures which can be used in implementing a graphic language comprise a separate subject, which is of sufficient importance and complexity to have warranted many studies in itself. But as Kulsrud stated, "Although the problem of data structure is a central one for graphics, it should not affect the graphic language design directly." Data structures are not a part of the graphic

language itself, but rather a component of a graphic system which is needed to implement the graphic language. Kulsrud further asserted that a complete graphic system would probably contain two or more different data structure types [15]. The Sketchpad system also makes use of the many buttons, knobs and toggle switches on the TX-2 computer. Although the order required in the selection of these hardware input devices determines the syntax of the dynamic graphic language used, the hardware and its input devices are not part of the actual language. Thus, when a graphic system is referred to, it implicitly determines a graphic language, but also includes additional components used to implement the language. Because of this, graphic systems are commonly referred to as graphic languages, and different graphic systems and graphic languages are often compared.

In order to clarify the situation, the author has supplemented a dictionary definition of language to define a graphic language for computers. The definition is the following: "a set or system of symbols or operations which can be used in a more or less uniform fashion to describe, generate and manipulate graphic displays on an input/output device which utilizes a digital computer to accomplish the necessary processing." This definition is felt to be adequate, but it is recognized that many other suitable definitions could be written.

C. A GENERAL PURPOSE VERSUS SPECIAL PURPOSE GRAPHIC LANGUAGE

There is some controversy whether one general purpose graphic language or many special purpose graphic languages are needed to accomplish the many and varied applications for computer graphics. Some leading scholars in the field believe that a specialized language is required for each application area if the graphic language is required for more than just drawing pictures [17]; [16]. Others feel that a language of utmost generality should be developed that permits its own modification [18]. There is, however, general agreement that a general purpose graphic language should have the capability of accomplishing more than just drawing a picture.

Since the uses for computer graphics are only limited by the imagination of man, any general purpose language, which was expected to be all-inclusive, would have to be used in the areas of computer-aided design, in drafting, in the design and analysis of electronic circuits, in the analysis of structural engineering, in numerical control in manufacturing, in the field of simulation, in the interpretation of pictures and the list would continue to grow on and on.

Notely stated that any display can be drawn theoretically by just three basic drawing commands (draw, rotate and move), but quickly adds that for most applications this method may be too cumbersome [19]. So, much more than just drawing

pictures is required and the applications are so varied that any one language that attempted to handle all applications would have to be either as basic as machine language, which is too cumbersome to use, or contain so many commands or procedures that it would take an extremely large storage capacity to implement. For example, Streit's VIP system which was designed only to draw displays took over 27,000 60-bit words to implement.

In the present state of the art, any so called high-level programming language, which is considered to be general purpose because it was designed to handle so many different areas of applications, such as PL/I, can not be efficiently or easily used for list processing, simulation and other specialized applications. Specialized languages have been developed to handle these more specialized applications. When a program is processed that utilizes only a small subset of the PL/I compiler, the efficiency, in regards to both time and storage of the utilization of the computer, is low. This is due to the fact that the PL/I compiler requires a larger amount of storage and takes longer to compile than many less complex compilers. All of the so-called general purpose, high-level programming languages, do have a common basic subset of capabilities which include data description and data transformation. In a similar manner, no one graphic language can be sufficiently general purpose to handle all applications. However, any general purpose graphic language (general purpose in the sense that the language can be used

with many varied applications) must have a basic set of required capabilities. To this subset of required capabilities, additional sets of supplementary capabilities are added, depending upon what specialized applications the language must handle. The required subset and these supplementary subsets then make up the graphic language. (See FIGURE 1.)

D. THE ATTRIBUTES AND CAPABILITIES OF A GENERAL PURPOSE GRAPHIC LANGUAGE

There are many different attributes and capabilities of a general purpose graphic language. The required subset of capabilities comprise those that are required while the supplementary subsets comprise those that are optional. The use or uses for which the language is designed is what determines what capabilities are included as optional.

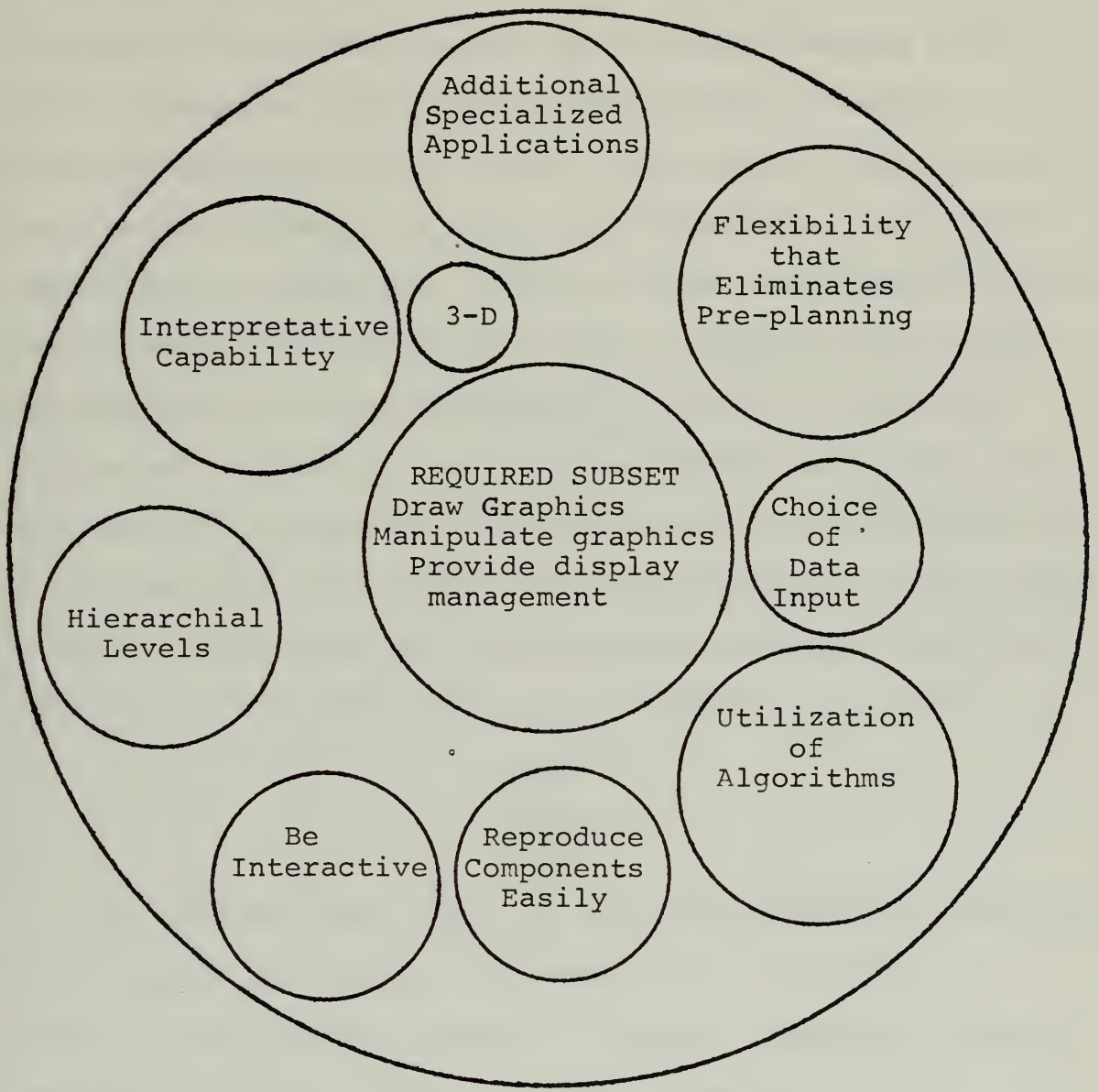
The basic requirements of a general purpose graphic language are:

(1) Draw graphics

(2) Manipulate graphics

(3) Provide display management

The most elementary requirement of a general purpose graphic language is that the language must describe and generate displays. In order to do this, the language must provide a capability of controlling the placement and intensity of points, line segments and possibly arc segments. In generating displays there are several different primitives which have to be considered. The basic building block



GRAPHIC LANGUAGE

FIGURE 1

is the line segment which is referred to as a line. Since the user of computer graphics is not beyond making a mistake, some means of correcting the mistake is needed. A complete blanking of the display with a start from scratch requirement is much too harsh and unforgiving, so an erase capability is mandatory. Titles, labels and numerical values are often used to amplify and explain displays so a means of entering text into a display is required. Since arcs, curves and circles are used frequently and generating them becomes such a tedious programming task, an arc function is a courtesy that the author feels every graphic system should provide. Therefore, the needed primitives which should be included in the subset of required capabilities are:

- LINE
- ARC
- ERASE
- TEXT

The language must give the user the ability to manipulate the displays by rotation and translation. In order to rotate a display efficiently, a rotation capability should be provided by the language. It is possible for the programmer to construct a program that accomplishes the rotation by drawing the component of the display in its new rotated position or, in the case of dynamic rotation, in incremental positions until the final position is reached. However, this defeats the purpose of a high-level programming language, which is to assist the user by easing the programming burden. A translation capability should also be provided

for the same reason. Both translation and rotation require some axis or point from which the objects can move or rotate. Some type of anchor point or reference point must be established, either by default or by selecting some specific point. Therefore, the graphic language should have the capability of selecting and changing this reference point. The required manipulators are:

ROTATE
TRANSLATE
REFERENCE POINT

In order to properly and efficiently handle displays and their components, the language must provide for the rearrangement, merging, searching and sorting of the components of a display. A means of retaining and recalling more than one display is required for convenience and completeness. In order to provide a minimum capability in display management the following processes or functions are required:

STORE
FETCH
NAME

These capabilities are the functions that form the required subset of any general purpose graphic language and as such are just the basic necessities of the language.

The main optional attributes and capabilities of a general purpose graphic language are:

(1) Be interactive

(2) Provide hierarchial levels within the display

(3) Provide an easy method of reproducing components within a display

(4) Provide a flexibility that eliminates extensive preplanning

(5) Provide an interpretative capability

(6) Provide for the utilization of algorithms

(7) Provide a choice of data input

(8) Provide a three-dimensional capability

For most applications, one of the most desired attributes of a general purpose graphic language is that it be interactive. For example, computer-aided design, which can include all types of creative design, is one of the most popular uses for computer graphics. In order to efficiently utilize computer graphics in this manner, it is necessary for the graphic language to be interactive. If the language is not interactive, the creativity of the human is greatly limited. The required time lapse between input, result and input will cause the user to lose his concentration on the subject. In many cases the user might be required to work on a different problem or application between inputs in order to efficiently utilize his time. To be truely interactive, the language must not only have a rapid response, but must also be forgiving if the user commits an error. If the remedial action required is too complex or time consuming, the interaction between the user and the system will be lost. The language can not be too complex or hard to learn for the same reason. The language should be open-ended (its capabilities easily extended). If the user arrives at a situation which he can not handle explicitly

within the capabilities of the language, there should be a means to extend the language through some programmable feature which should allow the user to maintain continuous thought while he implements the necessary extension.

The language should establish hierarchial levels among the various component parts of the display. This permits the user to allow the various levels to have specific characteristics (e.g., dynamic rotation of all components at one level or a specific intensity setting for all components below a component at a higher level). This capability would permit a display to depict a vehicle, which has wheels of two different sizes, in motion. The different sized wheels would be rotating at different speeds while the vehicle moved across the display. The number of levels that should be permitted is a moot point, but most scholars are in agreement that the language should be multi-level [9;15].

It is necessary for the general purpose graphic language to have an efficient method of reproducing components of a display since many displays contain elements which are similar except in size and/or location. When dealing with hardware which is limited to straight line segments, it is often necessary to approximate a circle with hundreds of small line segments. For the user to do this each and every time a circle is required, is a very inefficient method. The most efficient way of reproducing components is normally

through subroutines or procedures; therefore, it is necessary that a general purpose graphic language have this capability.

An important capability to a graphic language is that of providing enough flexibility within the language to eliminate any extensive prior planning. The present batch-type operating system in normal computer processing requires complete prior planning. This is not desirable and can not be accomplished in many computer graphics applications. As Chen and Dougherty state, "In interactive problem solving, unanticipated situations frequently arise that make complete preplanning difficult or impossible." [20] Gaglians and his co-authors assert:

> "Thus, effective use of graphics devices for interactive problem solving requires some means for requesting that a data processing system perform functions not anticipated at the beginning of the problem solving process." [7]

The flexibility is needed because in most cases the user will create or design some new display. In many cases the user will have very few preconceived ideas and will tax the imagination in creating a display. The freedom from pre-planning every construction is critical in interactive computer graphics. The user often does not know exactly what steps to take while creating a display so the user can not always create a normal type program prior to the execution phase. One way of providing this flexibility is to permit the user to return to a common point in the processing

which allows the branching to many, if not all, the functions or processes provided by the language.

Some graphic languages restrict the user to only constructing displays, (e.g., DRAWL [21] and VIP [9]). This restriction severely limits the use of these languages. As Roberts stated,

> "However, the ability to make pictures is not sufficient in itself; the pictures must be representative of data which needs computation such that the graphics system is used as an input/output tool, not merely as a display."[17]

By having the capability to interpret displays the computer can be utilized more effectively. There are several different ways a display can be interpreted. The most obvious is by topological analysis (i.e., analyzing the relation of one subelement of a display with respect to another). Another type of analysis is that which examines and locates special features in pictures (which the present state of the art normally handles through a digital photograph scanner [15]. It is desirable that a general purpose graphic language have the capability to handle elementary topological analysis from which more sophisticated analysis can be programmed.

Most users desire the capability of specifying algorithms in order to provide a more dynamic flexibility in the operation of the display console. By having this capability the user can use conditional statements, do-loop sequences and arithmetic statements. The capability is also important when the user is programming the display device primarily as an output device.

Some languages require certain data inputs only through a light pen or other attention devices, while others require the same data input solely through a keyboard device. Often the type of input device or devices are limited by the funds available to the computer installation for purchasing hardware. In other cases data may be more easily entered by one means than another. Therefore, it is desirable that the language permit the user the option of choosing the desired method of input.

Numerous uses of computer graphics are more suited to three-dimensional viewing than two-dimensional viewing. The work of Johnson, as demonstrated by Sketchpad III [22], and Roberts [23] have shown that it is possible to effectively use computer graphics in a three-dimensional representation. Rotation, magnifications, translation and perspective transformations can be accomplished by a single 4 X 4 matrix developed by Roberts [23]. The implementation of the three-dimensional aspects of a graphic language is a subject worthy of a complete study in itself. For extremely complex displays, such problems as determining hidden lines are too costly in computer time and storage to make three-dimensional displays practical. These problems can be circumvented by using wire frame displays or making some other compromise. If a three-dimensional capability is needed additional capabilities should be provided. Hidden or invisible lines should be available to the user to fully develop a display. This capability provides the user with the ability to have

objects appear as they do in real life (with their hidden lines), yet any analytic routines can operate on the complete object. A dash-line function often assists the viewers of a three-dimensional display to get a proper perspective because the user can dash the hidden lines. Three-dimensional representation has proven not only to be very effective, but also to increase greatly the creativity of the user.

These optional capabilities form supplementary subsets which can be added to the subset of required capabilities as needed. Other specialized capabilities can also be added for more specialized applications. (See FIGURE 1.)

E. A COMPARISON OF VARIOUS GENERAL PURPOSE GRAPHIC LANGUAGES

A comparison of some of the more interesting general purpose graphic languages is shown in Table 1 and is amplified in the following paragraphs.

1. Sketchpad

Although Sketchpad was created in 1963 it has many features that few, if any language, explicitly provides. At the touch of a button, lines can be made perpendicular, parallel, or be manipulated to meet other constraints. A "lock on" feature is provided that permits the user to terminate a line segment exactly upon intersecting another component. Pictures and subpictures provide hierarchial levels for the system. Points can be designated as attachment points on subpicture components; moreover, the

	ARC or CIRCLE	ROTATE	COPY	HIDDEN LINES	HIERARCHY	TOPOLOGICAL ANALYSIS	PICTURE ANALYSIS	CONSTRAINTS	HARD COPY	EXTENDABLE	COMPLEMENTARY LANGUAGE
SKETCHPAD	E	E	E	NO	2	M	NO	E	E	NO	NONE
GSP	P(L)	P(L)	P(L)	P(L)	P(L)	M	NO	P(L)	P(L)	M	FORTRAN
GRAF	P(L)	P	P(L)	E	P(L)	M	NO	P(L)	P(L)	NO	FORTRAN
GRAPHSYS	E	E	E	E	10	M	NO	P(L)	E	NO	AED MAD
IGS	E	P(L)	P(L)	P(L)	P(L)	M	NO	P(L)	E	NO	FORTRAN PL/I
VIP	E	E	E	NO	3	NO	NO	P	E	E	NONE
KULSRUD	E	E	E	NO	2	E	E	P	E	E(L)	FORTRAN MAD

SYMBOL LEGEND

E- Explicitly provided by the language

P- Capable of being programmed

M- Capability provided to a minimal degree

(L)- Provided by or in conjunction with a complementary language

2, 3, or 10- Number of hierarchial levels established

NO- Language does not have the capability

TABLE 1.

components can be joined at these selected positions. A component can be copied at the touch of a button. These are just a few of the many sophisticated features provided by Sketchpad. Sketchpad is a complete graphic system. It was specifically designed for the TX-2 computer. The input statements are completely hardware dependent and the language established by these statements can not be extended without a major change to the system. Sketchpad provides a graphic system with many unique features to the user. But even more important is the fact that it established the feasibility of computer graphics to the world [10].

2. Graphics Subroutine Package (GSP)

GSP consists of some basic subroutines which give the user a very elementary graphics capability. A simple program which does nothing but construct an arc becomes a relatively complicated task in GSP. Since GSP is designed to be used with FORTRAN, most graphic functions are obtainable by brute force programming. More recent developments permit GSP to be used in conjunction with COBOL or PL/I. The general form is CALL NAME (PARAMETER1, PARAMETER2, etc.) which is quite unwieldly when many parameters are required. GSP with the usual version of FORTRAN is more effective as a language that uses the display screen as an output device since all input device signals must be anticipated when the FORTRAN program is written. As previously discussed, many applications of computer graphics can not be pre-planned, so in many cases the input device signals can not be

anticipated. This drawback to GSP can be over come with the use of an incremental compiler or interpreter as was done with DISPLAYTRAN [7]. With interpretive FORTRAN execution, the attention device signals can be anticipated as the need for their use occurrs and no extensive pre-planning is required [24].

3. GRAF

GRAF provides basically the same capabilities provided by GSP with the exception that GRAF is an extension of FORTRAN, thus used exclusively with FORTRAN. One advantage of GRAF is that subroutine calls with their many parameters are avoided. As the authors of GRAF state, "Further, we feel that coding, debugging and simply understanding the logic of a program from its listing are all made much easier by avoiding CALL statements with long argument lists for frequently needed graphic routine." [3] Both GRAF and GSP were an attempt to ease the burden of programming on the graphics user. Since FORTRAN was probably the most commonly used programming language at the time, it was felt that by allowing the user to program in FORTRAN it would be easier for him than requiring him to learn a completely new language for graphics. An incremental compiler or interpreter should be used with GRAF because the same problem arises handling attention signals in GRAF as was described for GSP (i.e., the attention signals must be anticipated) [3].

4. GRAPHSYS

GRAPHSYS is a set of procedures or subroutines which is written in AED. Although GRAPHSYS was specifically designed for use at the Electronic Systems Laboratory, MIT, it is not as hardware dependent as Sketchpad since AED is a machine-independent language. GRAPHSYS could be implemented without many major modifications at other installations with adequate computer hardware. GRAPHSYS is part of a larger time-sharing system which is the mode that many graphics systems will use in the future. GRAPHSYS has many interesting features which are intended to ease the programming burden on the user. These include specific functions to accomplish such things as drawing a circle, making a copy and constructing hidden lines. A hierarchial level is provided among the components of a display. The language permits a depth of ten levels referred to as subpictures. These give the user a great deal of flexibility in constructing his display. GRAPHSYS was specifically designed to handle three-dimensional graphics so hidden lines and other functions needed with a three-dimensional display are available [4].

5. Integrated Graphics System (IGS)

IGS is a graphic system which is hardware independent, although implemented on the IBM 2250 graphic display console. It can be used with any language which has OS/360 linkages (e.g., FORTRAN, PL/I, Simscript 1.5 and OS/360 assembly languages). IGS is composed of many procedures or subroutines designed to handle the graphic functions necessary in

creating and manipulating the graphic displays. Calls to IGS routines are made from within the user's program. Parameters are handled either by the normal passing with the call or by using a special parameter array (200 locations). This array contains what could be considered the default values of the parameters in question. Since IGS provides only the elementary graphic functions, a user is required to write a rather complex program to construct even simple displays. For example constructing a simple circle would be a tedious task. An incremental compiler or interpreter should be used because attention device signals have to be anticipated or they will be ignored as was the case with GSP and GRAF [8].

6. Visual Interpretive Processing (VIP)

VIP is also a complete graphics system which was designed solely to draw displays; therefore, it does not qualify as a true general purpose graphic language. It is an interesting graphic system because it allows almost complete flexibility to the user. Little or no programming experience is required to use the system; yet more sophisticated programs can be constructed through "programmed functions" which are developed by the user at the display console. The complexity and sophistication of these programmed functions depends on the programming expertise of the user. A function interpreter carries out the execution of both the programmed and system functions. The interpreter fetches the code of the programmed functions into core,

permits nested functions by utilizing a stack, and handles various error conditions (i.e., infinite looping and illegal addressing). Attention device signals can be handled as they occur, which eleiminates the requirement to pre-plan the signals. This technique gives the user great flexibility in designing a display. The system is relatively hardware independent and provides two hierarchial levels. An algorithmic program can be developed through the programmed function capability [9].

7. Kulsrud's Model Language

Kulsrud's model graphic language is felt to be a true, general purpose language, designed to describe, generate, manipulate and analyze displays. In Kulsrud's article, he discussed only a typewritten version of his language, but he explained that this is done for convenience and to facilitate understanding. He states that this version could be translated to suit the graphic equipment available using light pen and control button sequences. Kulsrud included the basic statements necessary to conduct both topological and other forms of picture analysis. He did not design his language to be used with three-dimensional displays which is a limitation. Kulsrud used a metacompiler, which used incremental compilation, to produce interactive graphic programs. This permits immediate testing of language syntax on a line by line basis and the immediate detection of most typographical errors. Kulsrud's language has three

hierarchial levels and was designed to be used in conjunction with the normal high-level programming languages, FORTRAN and MAD [15].

F. DISCUSSION OF SYNTAX VERSUS SUBROUTINE APPROACH TO GRAPHICS

In the previous discussion of graphic languages and their implementation, there were basically two approaches used to implement the languages. Either a series of subroutine calls with their required parameters are made to graphic procedures stored in a library, or a syntax for the graphic language is specified and then the language is complied or interpreted by standard techniques. In the latter case some programs are compiled as an entire program while others are compiled line at a time by an incremental compiler. Since an interactive mode is desired, compilation should be on a line at a time basis. This was the method used in Kulsrud's graphic system [15]. When using subroutines to accomplish the graphic functions, some systems compile the subroutines prior to storing them so they are available in machine code for execution as the user desires. Other systems store the subroutines in their high-level language and then compile the routines with the entire program as they are called. The more desirable method is to compile the subroutines prior to storing them in order to decrease the time required to execute the procedure. This capability may involve dynamic loading with its overhead.

When considering the user inputting statements and/or data via the display console, either method has the capability to utilize attention devices (e.g., light pen, Rand tablet, mouse, etc.) to a maximum, keeping typewritten inputs to a minimum. It is usually more natural and quicker to point a device at a location on the display to determine the position of a point, rather than calculating the desired coordinates and then typing the coordinates into a typewriter. If the attention devices are used to the maximum, it is irrelevant to the user, inputting the information via the display console, whether the subroutine or syntax approach is being used.

If, on the other hand, the user is inputting statements and/or data via some non-graphical input device, which method used does become of interest. As previously mentioned, using subroutine calls with the many required parameters is quite unwieldly at times. However, the subroutine approach is usually more easily extended than the syntax method. Normally a subroutine can be programmed and placed in the system library more easily than the incremental compiler can be changed in the syntax method. This drawback to the syntax approach has been largely overcome by the metacompiler which makes the necessary changes to the graphic compiler as implemented by Kulsrud [15].

The subroutine approach is often more flexible because the subroutines may be used with many different languages. In considering the more recent syntax type graphic languages,

some of them have been designed to be implemented in conjuction with several high-level programming languages.

Graphic languages, in general, can be specified by their syntax as demonstrated by Morrison [16] and others. The syntax approach normally has a smoother program flow than that of the subroutine method. Both approaches have advantages and disadvantages; thus, the determination as to which method should be used should be decided on an individual basis for each computer installation. At an installation where the core memory is limited to such a degree that only a very carefully selected subset of a general purpose language can be implemented, the subroutine approach has a decided advantage. The selected subroutines can be implemented in a very basic language (i.e., assembly language or machine language); therefore, no large amount of storage is required for an incremental metacompiler, incremental compiler or interpreter as is the case of the syntax approach. Even if the sophisticated compiler is to be paged in and out of core memory, the increased complexity of the resident monitor will increase the storage required by the monitor which reduces the core memory available for the user. In the case of a large computer graphic installation (at least large in storage capacity), a syntax approach with its algorithmic-programming capability has an advantage because of the smoothness and flexibility that this method provides.

In comparing the installation where each approach could be implemented, the syntax method is the more difficult. Normally a system using the syntax method will require the services of a system programmer in order to program and maintain the required software. The subroutine approach can usually be programmed and maintained by the user so there is no necessity to hire a system programmer. The speed of execution is normally greater in the subroutine approach since the subroutines can be compiled into machine code and stored prior to execution.

III. GPGL, A MODEL INTERACTIVE, GENERAL PURPOSE GRAPHIC LANGUAGE

GPGL contains both the attributes which are required of a general purpose language and many of those that are optional. It includes the option of selecting two-dimensional or three-dimensional displays, the option of attention device inputs or keyboard device inputs, and the capability of constructing algorithmic programs. In addition, the language is intended to be conversational (i.e., every users action is met with some action or response from the computer). As previously mentioned the model could be implemented in its entirety or in a selected subset. Since the language is designed to handle most applications, it would require a great deal of memory storage if fully implemented; therefore, it is envisioned that implementing a selected subset would be more practical for most computer facilities.

A. AN OVERVIEW OF GPGL

GPGL is designed to provide the user with two different types of functions with which the user can accomplish the desired tasks. These functions are called system functions and user-defined functions. The system functions are provided explicitly within the language (e.g., rotation, translation, etc.), while the user-defined functions are designed by the user to accomplish the specific process or processes desired. The user-defined functions are normally

programmed by the user through teletype-system and teletype-editor commands. The functions are built by using the many available system functions as the basic elements from which a program is constructed for each user-defined function. The program is compiled and the user-defined functions stored under a unique name awaiting call.

The user accomplishes the desired tasks by first selecting whether a two-dimensional or three-dimensional display (no mixed mode is permitted) is desired. Then the user selects a series of system and/or user-defined functions from a "menu" (a list of optional choices) shown on the display console. The choice of how many and what functions are selected is basically determined by the user (some ordering is required by the language in respect to the input mode - attention device or teletype) which gives the user the necessary flexibility usually required in computer graphics as previously discussed.

The language provides a tri-level hierarchial structure. The basic or lowest level is an image, which is a component of a subpicture. Subpictures in turn compose or form a picture. Theoretically, there is an unlimited number of images in a subpicture and an unlimited number of subpictures in a picture. In actuality the number of either is limited by the storage capacity of the hardware available and the actual restrictions created by the implementation of the language. These images, subpictures and pictures can be uniquely named, and then stored and retrieved

through their name. Although it is envisioned that three hierarchial levels should be sufficient for most applications, the language could be extended in hierarchial depth without too much difficulty. GPGL could be extended in a similar manner to that used in GRAPHSYS [4]. This extension, however, would decrease the available user's storage capacity because of the need to store the necessary pointers and directories required to extend the hierarchial capability.

To assist in the visualization of the hierarchial levels included in the display and to permit the user to protect portions of the display that have been completed, a foreground and a background is established within the display. The foreground consists of the images which have not yet been stored in a subpicture (foreground makes up the current subpicture) while the background is composed of the subpictures which at the time make up the current picture. The primitives can effect only the foreground which provides a degree of protection to the subpictures and picture composing the background. It is intended that when GPGL is implemented the background portion of the display appear with a lower intensity than the foreground to assist the user in visualizing the hierarchial levels.

GPGL was specifically designed for some type of system which uses the subroutine approach to computer graphics in the implementation of the language. Specific teletype system commands were included that would permit programs written in other languages to be entered if the compiler in

the system could compile the other language or languages involved and store them as subroutines. GPGL could be used with a system which uses the syntax approach to computer graphics with some appropriate changes. These matters depend on the actual installation and the specific system used to implement the language. See the discussion of the syntax versus subroutine approach to computer graphs (paragraph II. F.).

When the user selects the three-dimensional mode, changes in the method of entering certain types of data are required. Since all locations in the viewer's space (three-dimensional space which the viewer would see if real objects were observed) have three dimensions it is necessary to enter three coordinates instead of the normal two. Different viewing conventions are required than those used in normal two-dimensional displays because the display screen is two-dimensional while the objects in the display are visualized in three dimensions. A convention implemented by Johnson with Sketchpad III is used [22], which includes three orthogonal and one three-dimensional perspective view. (See FIGURE 2.) It is envisioned that the system implemented would allow the user to increase the size of any of the four quadrants to fill the entire screen when selected. The four quadrants are not four independent displays, but are all interrelated so that an arc being drawn in one view is displayed in the other three. GPGL was designed to be implemented by using the three-dimensional, homogeneous

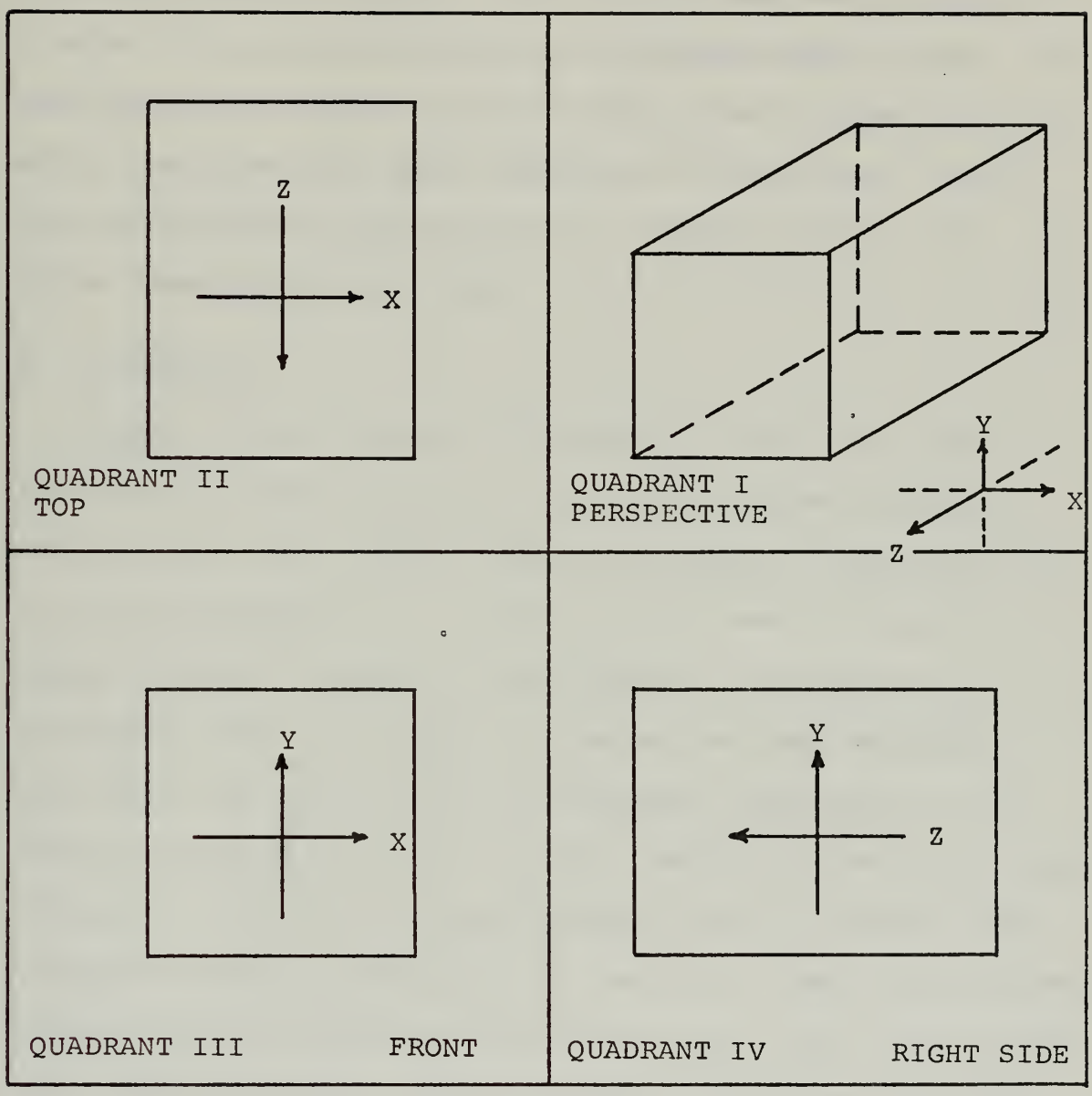

Z
X
QUADRANT II
TOP
QUADRANT I
PERSPECTIVE
Y
X
Z
Y
X
QUADRANT III
FRONT
Y
Z
QUADRANT IV
RIGHT SIDE

FIGURE 2

coordinate system developed by Roberts [23]. When entered in a typewritten-type mode, each point designated would be a series of four numbers. Two designations are required if a point is to be selected by an attention device input. A designation is made on the two appropriate orthogonal views which specifies the three dimensions of the point. This type of system was implemented by Johnson with his Pen Space Location Program [22].

B. FUNCTIONS

There are two levels of functions. The lower level functions or basic processes are called system functions and give the user control over each specific operation that is to be performed (e.g., draw a line, erase a line, etc.). These functions require no development or programming on the users part; the functions desired are just selected. This gives the user with no programming experience or expertise the ability to sit at the console and construct simple displays. The user-defined functions are the higher level functions and are programmed by the user. They are programmed through the user-defined instruction set. This set includes the system functions (whose arguments now become operands), the teletype system commands and the text-editor commands. The user-defined functions permit the use of an algorithmic approach in order to develop relatively complicated programs. User-defined functions can have either externally or internally specified operands which give the user a great deal of flexibility.

1. System Functions

System functions consist of primitives, manipulators, storage and retrieval functions, teletype function, and dimension selection functions. Functions which have a "-TT" suffix require teletype input or some other type of keyboard device input and the suffix is intended to act as a reminder to the user.

Most of the functions have parameters that can vary from points, to images, and in some cases to subpictures. A point can be located by several different methods. These methods are:

(1) selecting the point with an attention device input (two selections are required in the three-dimensional mode);

(2) by entering the X-coordinate, Y-coordinate (the Z-coordinate and the scale factor in the three-dimensional mode) through a keyboard device;

(3) by entering a unique name which has been previously assigned to the point.

In order to avoid ambiguity, subpictures and pictures are selected by name while images can be selected by selecting a point in the image or by name. The noun "component" can refer to any of the three hierarchial levels. In describing each system function, the function name will be followed by a verbal description of the parameters to be entered with the function. The necessary remarks explaining the function are included under the function.

a. Primitives

The primitive functions are the most basic of any of the functions and as such are used to construct and form the displays. The primitive functions can only be utilized in the foreground (at the image level) of the display. The primitives are as follows:

(1) Point Function

POINT (coordinates)

Remarks: POINT establishes a point with its coordinates as assigned in the inputs. The user has the option to continue to define additional points without having to re-select POINT.

(2) Line Function

LINE (end point coordinates)

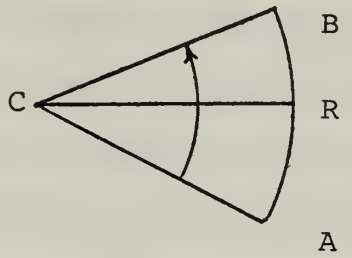
Remarks: Constructs a straight line segment joining the given end points. The coordinates of the end points are the parameters of the function. The user has the option to continue to draw lines by defining additional points without having to reselect LINE. Each additional point is the end point of a line segment from the previous point designated to the additional point last entered.

(3) Arc Functions

ARC (center (C), radius (R), delimiting point coordinates (A) and delimiting point coordinates (B))

Remarks: Constructs a circle segment (or circle) with the center at C and with line segment CR as the radius. The

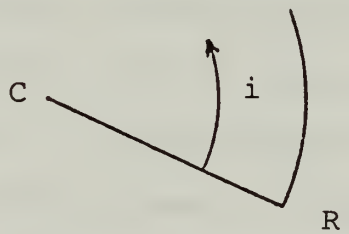
arc is determined by the angle subtended by the two line segments AC and BC as shown below:



If the two delimiting points, which are the third and fourth points entered, are omitted, a circle is drawn.

ARC-TT (center (C) and radius (R) point coordinates and the number of degrees (i))

Remarks: Constructs a circle segment (or circle) with the center at C and with line segment CR as the radius. The arc originates at R and extends through the number of degrees (i) entered in a counter clockwise direction as shown below:



(4) Text Input Function

TEXT-TT (coordinates of a point where the text is to be located, the size of the text desired and the string of text)

Remarks: Accepts the text message from the teletype and places it at the coordinates of the point entered. The system querys the user as to size of text and then requests the actual text message to be entered.

(5) Erase Function

ERASE (coordinates of a point in the selected image or the name of the image)

Remarks: Removes the designated image from the foreground display (releases or frees the storage previously utilized by the selected image so that the memory cells are available for use).

b. Manipulators

The functions that are used to manipulate and alter the images, subpictures and pictures are classified as manipulators. Some manipulative functions act upon either the current image, which is the image that is presently open for additions to its display list, or the entire subpicture, which is the foreground of the display. If the current image is still open, the manipulative function will act upon the image, if closed, the function will act upon the current subpicture, which is the foreground. (The storage and retrieval function NAME closes an image and is discussed in the next section.) The manipulative functions give the user the option of manipulating the current image or the foreground. If the user desires to manipulate the foreground, the current image must be closed. The manipulators are as follows:

(1) Reference Point Function

REF (coordinates)

Remarks: Designates the reference or anchor point. This reference point is the point which the image or foreground

will be manipulated around. The default value is the center of the display or in the case of the three-dimensional quadrant view, the center of each quadrant.

(2) Translation Function

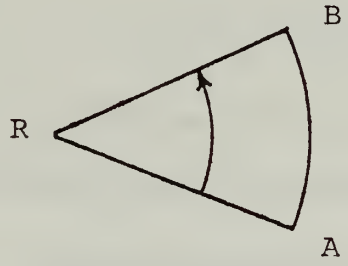
TRAN (coordinates)

Remarks: Translates the reference point from its previous position to the location entered. This causes the entire image or subpicture to translate the same distance and direction that the reference point moved.

(3) Rotation Functions

ROT (coordinates of two delimiting points (A and B))

Remarks: Rotates the image or subpicture about the reference point (R) in a counter clockwise direction through an angle subtended by the two line segments AR and BR as shown below:



(4) Flip Function

FLIP (factor)

Remarks: Reflects the foreground about a vertical axis through its reference point.

(5) Zoom Function

ZOOM-TT (factor)

Remarks: Magnifies the image of the subpicture by the

factor entered. If a negative number is inputed, the image or foreground is diminished (down to the limit of a point).

(6) Proportional Change In Size - X Axis

PROPX (coordinates of two delimiting points)

Remarks: Shrinks or expands the foreground symmetrically about a vertical axis through the reference point (RP) in the proportion:

$$(b_x - RP_x)/(a_x - RP_x)$$

where a_x is the X coordinate of the first point entered and b_x is the X coordinate of the second point entered.

(7) Proportional Change In Size - Y Axis

PROPY (coordinates of two delimiting points)

Remarks: Shrinks or expands the foreground symmetrically about a horizontal axis through the reference point (RP) in the proportion:

$$(b_y - RP_y)/(a_y - RP_y)$$

where a_y is the Y coordinate of the first point entered and b_y is the Y coordinate of the second point entered.

(8) Display Reference Point

DRP

Remarks: Displays the reference point of the image or subpicture as an asterisk.

(9) Dash Function

DASH

Remarks: Changes all the lines in the image or subpicture into a dashed representation.

(10) Hidden Line Function

HIDDEN

Remarks: Changes all the lines in the image or subpicture to an invisible mode. The lines are still present in the data structure even though the lines do not appear on the display console so interpretative functions can still be utilized with respect to the invisible component.

(11) Intensity Function

INTENSITY-TT (factor)

Remarks: Intensity of the image or subpicture is varied by the factor (within the limits prescribed by the hardware). If the factor is positive, the intensity is increased; and if it is negative, the intensity is decreased.

(12) Graytone Function

GRAYTONE

Remarks: Used to half-tone, that is, shade the image or subpicture. This capability has been successfully implemented as discussed by Sutherland [25].

(13) Scale Function

SCALE-TT (scale for X, Y, and Z)

Remarks: Defines the picture, with coordinate axes X, Y, Z, as being 2X units in width, 2Y units in height, and 2Z units in depth. The origin (0,0,0), resides at the center of the screen (center of each quadrant in the three-dimensional, quadrant view). The range is from -X to +X, -Y to +Y, -Z to +Z. This permits the user to use any scale desired whether it be miles, feet or whatever.

c. Storage and Retrieval Functions

In order to provide convenience and completeness to the user, various functions are needed to store and retrieve images into or from a subpicture, subpictures into or from a picture and pictures into or from a library. The storage and retrieval functions are as follows:

(1) Frame Component Function

FRAME

Remarks: Appends the current contents of the foreground display to that of the background display as in internal structure (subpicture). Automatically gives a unique subpicture number for retrieving the structure. (The user has the option of using the NAME function to give the structure a unique name). The intensity of the foreground is reduced in order to assist in the visualization of the hierarchial levels.

(2) Store Picture Function

STORE-TT (name)

Remarks: Stores the current contents of the background display in the storage area or library for pictures under the name entered.

(3) Retreival Function

FETCH-TT (name)

Remarks: Retreives the image, subpicture or picture, whose name is entered. If the component named is an image or subpicture it is displayed as part of the current foreground (the component must be a component of the current picture);

if the component name is a picture it replaces the present contents of the background display. Images and subpictures retreived are opened for the addition of vectors or other modifications.

(4) Name Function

NAME-TT (name)

Remarks: Assigns the name entered to the header (first location) of the designated image or subpicture (pictures are named by STORE-TT function). Each name must be unique to avoid ambiguity in retreiving the image or subpicture. In order to explicitly assign a name to a subpicture, NAME must be called immediately after FRAME. When NAME is used in regards to an image, it "closes out" the image. "Closes out" means that no additional vectors can be added to the display list of that image until the image is retrieved by the FETCH function.

(5) Delete Function

SCRUB-TT (name)

Remarks: Deletes the picture specified by the name from storage or the subpicture specified by the name from the background. Frees the storage previously utilized by the picture or subpicture which is scrubbed.

(6) Clear Foreground Function

CLRF

Remarks: Blanks the foreground display and frees the storage utilized by the images in the foreground display (in most cases the user will have stored the images desired for retention in a subpicture).

(7) Clear Background Function

CLRB

Remarks: Blanks the background display and frees the storage utilized by the subpictures in the background display (in most cases the user will have stored the picture desired in order to retain the subpictures and images).

(8) Hard Copy Function

PHOTO

Remarks: Generates a hard-copy of the entire console screen.

d. Keyboard Device Functions

When the user desires to input specific data which is normally entered by attention device through a keyboard device, the TTY function must be called.

TTY

Remarks: Alerts system that the normal input by attention device will be via a keyboard device. All inputs for the function must then be entered by a keyboard device. Functions which have the suffix "-TT" should not be followed by the TTY function unless the normal attention device inputs (if any) are to be entered via the keyboard device. For example, with ARC-TT if the first two delimiting points were to be entered by a keyboard device instead of an attention device the TTY function would be used.

e. Analysis Functions

Functions which interpret the topology and other pictorial features of a display are required in a general purpose graphic language. Certain basic analysis

functions are provided in GPGL which permit the user to develop more complex interpretative programs. Some functions return a value and print the value by teletype. If the function is used in a user-defined function, the teletype message is not printed.

(1) Within Function

WITHIN (coordinates of two points with each point designating an image, or two names designating images or subpictures)

Remarks: Checks whether the component entered first lies within the second component entered.

(2) Separate Function

SEPAR (coordinates of two points with each point designating an image, or two names designating images or subpictures)

Remarks: Checks whether the selected images or subpictures are separated. If there is no intersection of lines or points, and one component does not lie within the other, TRUE is returned and printed by the teletype, otherwise FALSE is returned and printed by the teletype.

(3) Simply-Connected Function

SIMPLY-TT (name of an image, subpicture or picture)

Remarks: Determines whether the designated component is a simply-connected region. (Simply-connected region is a region for which any closed curve lying in the region can be continuously shrunk to a point without leaving the region [26].)

(4) Region Assignment Function

REGSNAP-TT (property)

Remarks: Assigns each point in the display (picture) to a region, which has internally generated labels with the property selected by the parameter entered. This process is referred to as a region snap. Normally the property is color,(i.e., black, white and/or shades of gray [15]).

(5) Name Region Function

NAMEREG-TT (coordinates of a point, the property parameter and a name)

Remarks: Assigns the name entered as the name of the region, with respect to the property specified by the property parameter, which the selected point or component is in.

(6) Connection Functions

CONECT (coordinates of two points)

Remarks: Checks whether the two selected points are in the same region. If they are connected (in the same region), a TRUE is returned and printed by the teletype. If not, FALSE is returned and printed by the teletype.

CONECTBY-TT (coordinates of two points and a name of a component)

Remarks: Checks whether the two selected points are in the same region and whether the named component connecting the two points is in the region. TRUE or FALSE is returned and printed by the teletype.

(7) Adjacent Function

ADJAC (coordinates of two points or the names of two components)

Remarks: Determines the adjacency of the designated components. If the two components are in the same region a zero is returned and printed by the teletype. If the components are in adjoining regions, a one is returned and printed by the teletype. If otherwise, the number of regions plus one, intervening between the two regions containing the designated components is returned and printed.

(8) Intersection Function

INTERX (coordinates of two points or the names of two components)

Remarks: Determines whether the designated components intersect. If they intersect any where in the display, the intersection value for the property previously used in the region snap is returned and printed at the teletype, otherwise FALSE is returned and printed at the teletype.

f. Dimension Selection Functions

The user determines which display mode he is going to use, either a two-dimensional display or a three-dimensional display and then selects the appropriate functions. The dimension selection functions are as follows:

(1) Two-Dimensional Function

2-D

Remarks: User selects this function first if he is going to use the normal, two-dimensional display mode.

(2) Three-Dimensional Function

3-D

Remarks: User selects this function first if he is going to use the three-dimensional display mode.

(3) Three-Dimensional View Functions

QUADI

QUADII

QUADIII

QUADIV

Remarks: Enlarges the selected quadrant to full screen size on the display console. (Used in the three-dimensional mode only.)

(4) Quadrant-View Function

3-D VIEW

Remarks: Returns the display to a quadrant view. (Used in the three-dimensional mode only.)

g. Keyboard Mode Function

In order for the user to develop user-defined functions, it is necessary to enter a keyboard mode where teletype system and text-editor commands are inputed through the keyboard device.

TTYMODE

Remarks: Causes the keyboard mode to be entered.

GET (coordinates of up to four points)

Remarks: Used to enter attention device inputs during the execution of a user-defined function. Provides the capability of allowing external inputs, whose locations can not be determined by the user prior to the execution phase. (Used only with user-defined functions.) Permits the input of up to four points into pre-planned and allocated storage locations.

2. User-Defined Functions

The user-defined functions, which are constructed by the users, are subroutines written by the user, compiled and stored under a unique name and then executed when a user selects the function's name from the menu. Therefore, user-defined functions developed by one user can be used to advantage by any other user. This gives the system an excellent growth potential, limited only by the storage capacity of the function library. The user-defined functions can be stored in pages or other segments.

The functions are normally formed by using the user-defined instruction set, which contains the system functions, the teletype-system commands and the teletype-editor commands. The teletype-system and the teletype-editor commands are similar to the ones used by Streit [9]. The choice of using similar commands to those of Streit's was made after examining the languages and systems previously mentioned. Streit's teletype mode is more natural, easier to implement, and much simpler to use than those of the other graphic languages and systems. The capability of using external programs (written in a language acceptable to the systems' compiler) is an important addition to Streit's teletype system.

From the user's standpoint, the user-defined functions appear the same as system functions once written and compiled. All locations in a user-defined function are referred to a coordinate system local to the function.

This requires the user to use a scale instruction when the function locates primitives on the display screen. The local coordinate gives the system user great flexibility and freedom in applying user-defined functions (but requires that the implemented system map all data locations between the user-defined function and the system's display). The user-defined functions can have internally or externally specified operands which give the user the ability to define any needed locations or points at the time the function is formed or when the function is called. The format used to define the user-defined instruction set is as follows:

Label: OPCODE A;B;C;...

or

a;b;c;...

Remarks: A, B, C...are symbolic address labels which are local to the user-defined function and a, b, c... are numerical operands. The label portion of the instruction is formed by an identifier followed by a colon, while OPCODE is the operation code given in the user-defined instruction set. Parenthesis are used to show all the different variations of the basic instruction.

a. User-Defined Instruction Set

The user-defined instruction set contains instructions formed by using the system functions as the OPCODE with the functions inputs as OPERAND as shown below for the function LINE:

Label: LINE A;B

Remarks: Connects the points A and B to form a line segment, AB.

In addition to the system functions, the user-defined instruction set includes arithmetic, conditional and control instructions. These instructions give the user an algorithmic-type programming capability which allows more flexibility, especially in respect to using the interpretative functions.

The arithmetic instructions give the user the basic arithmetic operations required, which include assignment, addition and subtraction. These instructions can be used for all the coordinate values or for the individual coordinates. The individual coordinate values are shown in parenthesis. The arithmetic instructions are as follows:

(1) Label: SET A;B
(SETX
SETY
SETZ)

Remarks: Assigns B to A.

(2) Label: ADD A;B
(ADDX
ADDY
ADDZ)

Remarks: Adds the X, Y, Z components of points A and B and places the result in A.

(3) Label: SUB A;B
(SUBX
SUBY
SUBZ)

Remarks: Subtracts the X, Y, Z components of points B from A and places the result in A.

(4) Label: SWITXY A

(SWITYZ

SWITXZ)

Remarks: Assigns the X component of A the value of the Y component and the Y component the value of the X component. (Assigns the Y component of A the value of the Z component and the Z component the value of the Y component. Assigns the X component of A the value of the Z component and the Z component the value of the X component.)

The conditional instructions allow conditional branching which permits the user to transfer control if various conditions are met. The instructions are as follows:

(1) Label: NZX A;B

(NZY

NZZ)

Remarks: Tests the X component of point A, and if non-zero, transfer control to B, otherwise control is passed to the next instruction. The alternate instructions test the Y and Z component of point A respectively, if non-zero, they transfer control to B, otherwise control is passed to the next instruction.

(2) Label: ZRX A;B

(ZRY

ZRZ)

Remarks: Tests the X component of point A, and if zero, pass control to B, otherwise control is passed to the next instruction. The alternate instructions test the Y and Z component of point A respectively, if zero, they pass control to B, otherwise control is passed to the next instruction.

(3) Label: NGX A;B
(NGY
NGZ)

Remarks: Tests the X component of point A, and if negative, transfer control to B, otherwise control is passed to the next instruction. The alternate instructions test the Y and Z component of point A respectively, if negative, they transfer control to B, otherwise control is passed to the next instruction.

(4) Label: PSX A;B
(PSY
PSZ)

Remarks: Tests the X component of point A, and if positive, transfer control to B, otherwise control is passed to the next instruction. The alternate instructions test the Y and Z component of point A respectively, if positive, they pass control to B, otherwise control is passed to the next instruction.

The unconditional transfer instruction passes control to the designated symbolic address.

Label: GOTO A

Remarks: Transfers control to A.

b. Teletype-System Commands

The teletype system commands are used to create and manipulate the text and code of the user-defined functions. The format of the commands is:

COMD1 COMD2/FIELD.

The first two fields are the command portion, where COMD1 specifies whether the command pertains to a function, picture or an external program, which is entered as a user-defined function. COMD2 is the action that the command is to perform. The remaining portion, which is FIELD, is the argument for the command. The command and field portion are separated by a slash and the instruction is ended with a period. Some instructions have no COMD1 portion and/or argument so the command portion consists of only an action part and the FIELD portion may be blank. The slash and period are always required. The command portion may be abbreviated to the first letter of the two fields (the underlined character or characters in each instruction). Blanks are used as delimiters except between the command and its argument where the slash is the delimiter. The teletype-system commands are as follows:

(1) Definition Command

FUNCTION DEFINE/NAME.

Remarks: This command opens a user-defined function titled NAME by entering the text-editor mode. When the user has completed his function, the text and code are stored under the symbolic address, NAME in the function library.

(2) Modification Command

FUNCTION MODIFICATION/NAME.

Remarks: Fetches the text of the present user-defined function with the symbolic address NAME. Deletes the code and enters the test-editor mode.

(3) Purge Command

FUNCTION PURGE/NAME.

Remarks: Deletes the text, code, and entry points for the user-defined function NAME.

(4) Change Name Command

FUNCTION NAME/OLDNAME NEWNAME.

Remarks: Changes all the entry points associated with the user-defined function OLDNAME to NEWNAME.

(5) Fetch Command

FUNCTION FETCH/NAME.

Remarks: Fetches the text for the user-defined function NAME.

(6) Fetch Code Command

FUNCTION CODE/NAME.

Remarks: Fetches the code for the user-defined function NAME.

(7) List Functions Command

FUNCTION LIST/.

Remarks: Lists all the user-defined functions in the function library.

(8) List Picture Command

PICTURE LIST/.

Remarks: Lists all the pictures in the picture library.

(9) External Program Input Command

PROGRAM INPUT/NAME.

Remarks: Accepts programs as inputs through paper tape, punched cards or other input devices acceptable to the implemented system. These programs can be in machine code or any high-level programming language which the system's compilers can compile. These programs are compiled and stored as a user-defined function executable on call. (Modification and Fetch system commands can not be executed on the external programs, which are in some other programming language.)

(10) Exit Teletype Mode Command

RETURN/.

Remarks: Teletype mode is exited and the user is returned to the function menu selection mode.

c. Text-Editor Commands

The text-editor commands permit the user to construct or modify the user-defined functions. The text-editor mode is entered by executing either the definition or modification teletype commands. The command format for the text-editor commands is similar to that used for the teletype-system commands. The format consists of:

COMD1 COMD2 COMD3/TEXT.

where COMD1 is the action indicator and COMD2 and COMD3 are the arguments. TEXT is the text lines of the user-defined function and consist of a label portion (LABEL:), an OPCODE portion and an OPERAND portion, which contains the arguments

of the functions in the OPCODE. Since some commands have only one argument or no arguments, COMD2 and COMD3 may be blank. Some TEXT lines have no label, so the label portion may be blank. If, instead of entering a text line, an existing text line is manipulated, the TEXT portion is blank. Blanks are used as delimiters except between the command portion and TEXT where the slash is used. The slash and period are required for all instructions. The text-editor commands are as follows:

(1) Next Text Line Command

NEXT/TEXT.

Remarks: Enters TEXT as the next line of the text-editor display.

(2) Insert Text Line Command

INSERT a/TEXT.

Remarks: Enters TEXT as the line above line a and below line a-1.

(3) Purge Text Line Command

PURGE a/.

Remarks: Deletes line a, and moves all the lines a+1 and greater up one line.

(4) Move Text Line Command

MOVE a b/.

Remarks: Deletes text line a and moves all the lines up one line and then inserts same text line, which was removed, above line b.

(5) Replace Text Line Command

REPLACE a/TEXT.

Remarks: Replaces the line a with TEXT.

(6) Compile Command

COMPILE/.

Remarks: Compiles the text displayed and loads the text and code into the function library under the name associated by the definition or modification teletype system commands. The system returns from the text-editor mode into the normal execution mode.

d. Examples of a User-Defined Function

In order to demonstrate the procedures required in constructing a user-defined function, two examples are discussed. The first example describes those teletype-systems commands and text-editor commands, which would be utilized in constructing the function. The second example shows only the text lines which make up a user-defined function in order to demonstrate the finished product.

The commands and the sequence in creating a user-defined function named HORIZON, which takes a given line and creates a horizontal line with the same X coordinates for its end points, are shown below with amplifying comments. The system functions are designated as (SF); the teletype-system commands are designated (TS); and the text-editor commands are designated (TE). The abbreviated format for the teletype-system and the text-editor commands is not used for clarity. A carriage-return character, which

signifies the end of each instruction, is not shown. The lines three through fifteen are used to construct the text lines which will accomplish the actions discussed in the comment portion when the user-defined function is executed.

NUMBER	COMMANDS	TYPE	COMMENTS
1	TTYMODE	SF	User selects the teletype mode with an attention device input.
2	FUNCTION DEFINE/HORIZON.	TS	Enters the text-editor mode with the name HORIZON, which is associated with the function to be constructed.
3	NEXT/ 2-D.	TE	Selects 2-D representation.
4	NEXT/ SCALE 500;500.	TE	Determines the local scale of the function to be 500x500.
5	NEXT/ CLRF.	TE	Clears the foreground.
6	NEXT/ GET A;B.	TE	Accepts two attention device inputs upon execution and loads their coordinates into the OPERAND portion of locations A and B
7	NEXT/ REF A.	TE	Moves the reference point to point A.
8	NEXT/ LINE A;B.	TE	Draws a line segment from point A to point B.
9	NEXT/ SETX TEMP;B.	TE	Assigns the X component of B to the X component of point TEMP.
10	NEXT/ SETY B;TEMP.	TE	Assigns the Y component of point A to Y component of point TEMP.
11	NEXT/ ROT B;TEMP.	TE	Rotates the image (line AS about A through the angle B-A-TEMP.*
12	NEXT/ RETURN.	TE	Returns control from the subroutine HORIZON.
13	NEXT/A POINT.	TE	Creates the symbolic address A and designates it as a point.

14	NEXT/B POINT.	TE	Creates the symbolic address B and designates it as a point.
15	NEXT/TEMP POINT.	TE	Creates the symbolic address TEMP and designates it as a point.
16	COMPILE/.	TS	Compiles the function HORIZON and stores the text and code in the function library under HORIZON and returns control from the teletype mode.

* The line segment AB is horizontal, but the length has been changed.

The text and amplifying comments for a user-defined function named PARALLEL, is shown below. The user-defined function HORIZON is used. A carriage-return character which signifies the end of each instruction is not shown.

NUMBER	FIELD1	FIELD2	FIELD3	COMMENTS
1		2-D		Selects 2-D representation.
2		SCALE	500;500	Determines the local scale to be 500,500.
3		CLRF		Clears the foreground.
4		GET	A;B;C;D	Accepts four attention device inputs and loads them in OPERANDS of lines 12, 13, 14, 15.
5		SET	TEMP1;B	Sets point TEMP1 equal to point B.
6		HORIZON	A;B	Calls the user-defined function HORIZON, which makes line segment AB horizontal.
7		HORIZON	C;D	Calls the user-defined function HORIZON, which makes line segment CD horizontal.
8		NAME	PARA	Names and closes image (PARA) so that next command will act on foreground.
9		REF	A	Moves the reference point to point A.

10		ROT	B;TEMP1	Rotates the foreground (lines AB and CD) about point A through the angle B-Z-TEMP1.
11		RETURN		Returns control from the subroutine parallel.
12	A	POINT		Creates the symbolic address A which is a point.
13	B	POINT		Creates the symbolic address B which is a point.
14	C	POINT		Creates the symbolic address C which is a point.
15	D	POINT		Creates the symbolic address D which is a point.
16	TEMP1	POINT		Creates the symbolic address TEMP1 which is a point.
17	TEMP2	POINT		Creates the symbolic address TEMP2 which is a point.

C. EXAMPLES OF THE USE OF GPGL

The use of GPGL (implemented in an imaginary graphics system) is demonstrated in the following examples. The procedures are described by listing in chronological sequence the functions and inputs which would be utilized by a user if he were actually at a console programming. The displays are kept relatively simple for clarity and ease of comprehension. The attention device inputs which are required to locate components of the display (i.e., points, lines, etc.) are considered to be light pen hits (abbreviated as LP) and numbered sequentially (e.g., LP1, LP2).

Teletype inputs are shown in capital letters and underlined. The selection of the actual functions used is represented by the name of the function with no attempt to show what method of selection would be used (i.e., light pen picks, depressed function switch, etc.).

A two-dimensional display is developed to draw a geometric pattern which is named EMBLEM and then manipulated.

ORDER	FUNCTION	ARGUMENTS	COMMENTS
1	2-D		User selects the two-dimensional mode.
2	CLRF		Clears foreground (not required if already clear).
3	CLRB		Clears background (not required if already clear).
4	LINE	LP1;LP2; LP3;LP4	Draws a triangle from LP1 to LP2 to LP3 to LP4 (LP4=LP1).
5	NAME-TT	A C/R	Names the triangel A.
6	LINE	LP5;LP6; LP7;LP8	Draws a triangle from LP5 to LP6 to LP7 to LP8 (LP8=LP5).
7	NAME-TT	B C/R	Names the triangle B.
8	FRAME		Frames trianges A and B into a subpicture.
9	NAME-TT	TRIANGLES C/R	Names the subpicture TRIANGLES.
10	CLRF		Clears the foreground.
11	ARC	LP9;LP10	Draws a circle with LP9 as center and the radius the line segment from LP9 to LP10.
12	FRAME		Frames the circle.
13	STORE-TT	EMBLEM C/R	Stores the picture named EMBLEM, which consists of two triangles and a circle.
14	CLRB		Clears the background.
15	CLRF		Clears the foreground.
16	FETCH-TT	EMBLEM C/R	Fetches EMBLEM from storage and displays it in the background.
17	FETCH-TT	TRIANGLES C/R	Fetches subpicture TRIANGLES and displays it in the foreground.
18	ROT-TT	90 C/R	Rotates the foreground (both triangles) 90 degrees CCW.
19	FETCH-TT	A C/R	Fetches image A (triangle A).
20	DASH		Dash function is called and triangle A is dashed.
21	CLRB		Clears the background.
22	FRAME		Frames the dashed line triangle A and triangle B into a subpicture.
23	STORE-TT	TRI C/R	Stores the picture named TRI which is composed of triangle A, which is a dashed triangle, and triangle B both rotated 90 from the original subpicture TRIANGLE.

A three-dimensional display composed of a rectangular solid is constructed. FIGURE 3 shows the solid and the alphabetic designation of its corners, which are enclosed in parenthesis in the example.

ORDER	FUNCTIONS	ARUMENTS	COMMENTS
1	3-D		Selects the three-dimensional representation.
2	SCALE	10;10;10 C/R	Sets the scale for each axis from -5 to +5. (ABCD)
3	LINE	LP1,LP2;(A) LP3,LP4;(B) LP5,LP6;(C) LP7,LP8;(D) LP9,LP10(A)	Constructs the front (ABCD) of the solid.
4	GRAYTONE		Shades the front (ABCD) a lighter gray.
5	NAME-TT	FRONT C/R	Names the square (ABCD) FRONT and closes the image.
6	LINE	LP11,LP12;(A) LP13,LP14;(D) LP15,LP16;(H) LP17,LP18;(E) LP19,LP20(A)	Constructs the right side solid (ADHE).
7	GRAYTONE		Shades the side (ADHE) to lighter gray.
8	NAME-TT	SIDE C/R	Names the right side (ADHE) SIDE and closes the image.
9	LINE	LP21,LP22;(D) LP23,LP24;(C) LP25,LP26;(G) LP27,LP28;(H) LP29,LP30(D)	Constructs the top of the solid (DCGH).
10	GRAYTONE		Shades the top a lighter gray.
11	NAME-TT	TOP C/R	Names the top (DCGH) TOP.
12	LINE	LP31,LP32;(B) LP33,LP34;(F) LP35,LP36(E)	Constructs the lines BF and FE.
13	HIDDEN		Lines (BEF) are changed to the invisible mode.
14	NAME-TT	EDGE C/R	Names lines (BFE) EDGE.
15	LINE	LP37,LP38;(F) LP39,LP40(G)	Line (FG) is drawn.
16	HIDDEN		Line (FG) is changed to the invisible mode.
17	NAME-TT	REAR C/R	Names line (FG) REAR.
18	FRAME		Entire solid with its hidden line becomes a subpicture.

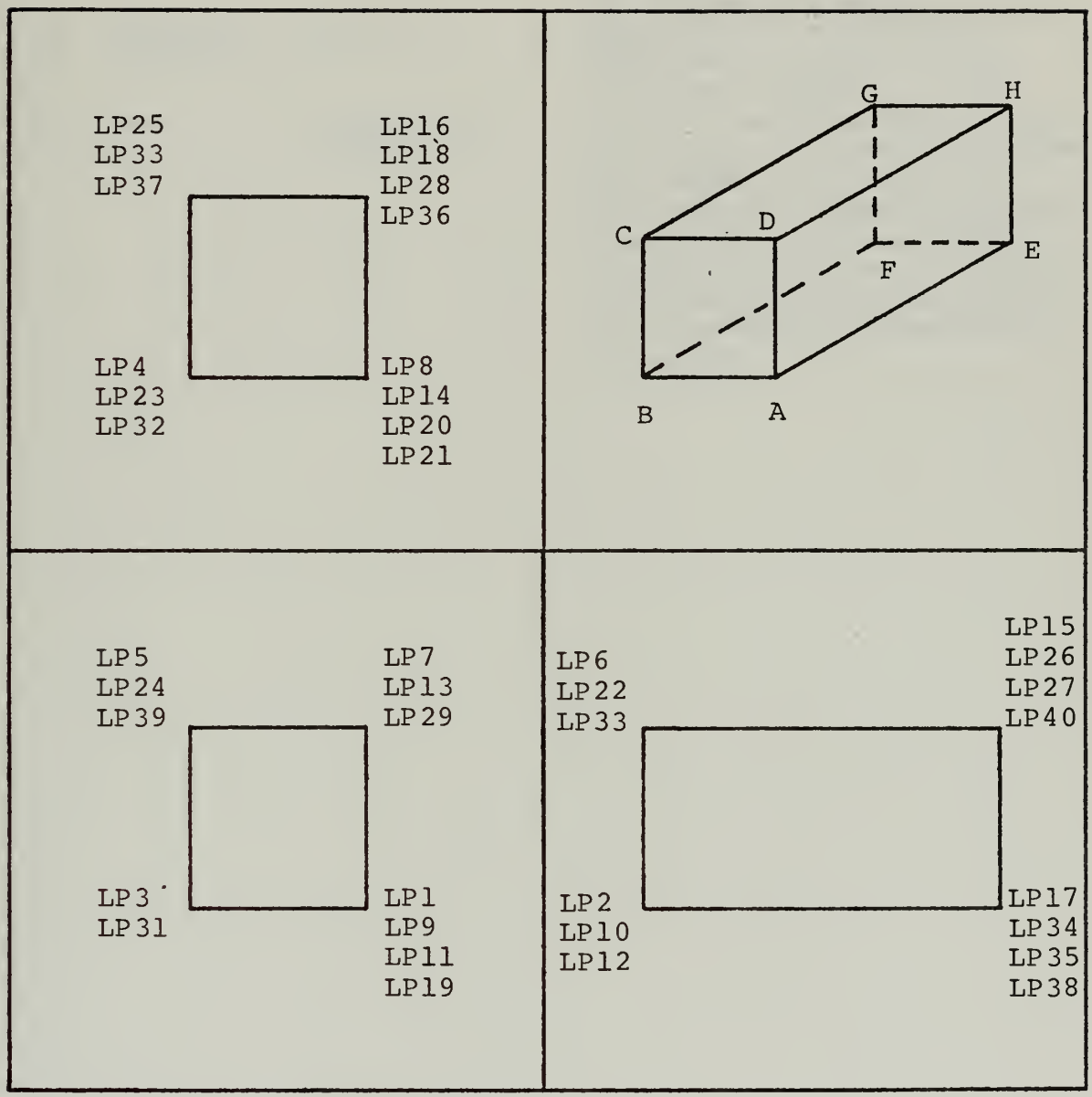

LP25
LP33
LP37
LP16
LP18
LP28
LP36
LP4
LP23
LP32
LP8
LP14
LP20
LP21
G
H
C
D
E
F
B
A
LP5
LP24
LP39
LP7
LP13
LP29
LP3
LP31
LP1
LP9
LP11
LP19
LP6
LP22
LP33
LP15
LP26
LP27
LP40
LP2
LP10
LP12
LP17
LP34
LP35
LP38

FIGURE 3

19	NAME-TT	BOX C/R	Subpicture is named BOX.
20	POINT		Point function is called.
21	TTY	2;2;-3;.75 C/R	Normal attention device input mode is changed to teletype mode and coordinates of X=2, Y=2, Z=-3 and scale of .75 are entered.
22	NAME-TT	Q C/R	Names point entered Q.
23	WITHIN		Calls the interpretative function WITHIN.
24	TTY	Q;BOX C/R	Usual attention device input mode is changed to teletype mode and Q and BOX are entered. Then the interpretative function WITHIN determines whether point Q is in BOX and prints either TRUE or FALSE through the teletype.

IV. IMPLEMENTATION OF GPGSY, A SUBSET OF GPGL, AT THE NPS

A. OBJECTIVES

GPGSY, a subset of GPGL, was implemented at the Electrical Engineering Computer Laboratory at the Naval Postgraduate School. The purpose was to illustrate some of the problems that are encountered in actually implementing GPGL into a graphic system. (The fact that a primitive graphics system, which is extremely easy to utilize and which will be easy to extend, is not ayailable at the NPS is only a by-product.) The primary objective in selecting the subset to be implemented was to examine the problems encountered in having a tri-level hierarchial language. The necessary pointers and directories to implement the image, subpicture and picture concept were of specific interest.

There are many reasons why a subset was implemented in lieu of the full GPGL language. The overriding reason was the impossibility of fully implementing GPGL with the hardware available at the computer laboratory. The fact that only a subset can be implemented is expected to be the rule instead of the exception for most computer installations. GPGL was specifically designed so that personnel at a computer installation can select a desirable subset, which both meets the needs and the capabilities provided by the available hardware. The selection of the subset at NPS tested this hypothesis.

GPGL was designed to be hardware independent and this was important since the subset had to be implemented on the specific hardware available at the laboratory. A desired capability for a general purpose language is that it be interactive; GPGSY presented an opportunity to see if at least a portion of GPGL was truely interactive. It was also desired to examine the feasibility of giving the user the option of entering data either by attention device signals or teletypewriter as permitted in GPGL.

B. THE IMPLEMENTED SUBSET

GPGSY is an interactive, general purpose graphics system which permits the user to construct two-dimensional displays on a cathode ray tube (CRT). (The hardware utilized was not designed for three-dimensional representation.) GPGSY requires a storage capacity of 1,843 30-bit words and is written in ADEPT, an assembly language. The ADEPT program with explanatory comments is appended to the thesis.

The system was implemented on an Adage Graphics Terminal, Model 10. The nucleus of the system is the Digital Processor, DPR2, which is a general purpose digital computer with a two microsecond memory cycle time and one microsecond register to register transfers. The core memory size is 8K with a 30-bit word length. A two pack disk drive is available for auxiliary storage. The graphics terminal consists of a cathode ray tube (CRT), teletypewriter, a vector generator, a character generator, a light pen and sixteen function

switches. A resident monitor (AMRMX) is used to store and retrieve programs from the disk pack, to process programs, and to control various system components. Portions of the monitor are explicitly used in implementing GPGSY by calling on it to receive and print teletype messages.

With the primary objective of examining the hierarchial levels in mind the primitive functions of LINE, (drawing a line, which is the basic building component) and ERASE (the capability to erase an image) were selected. LINE was, of course, necessary in order to construct a display, and ERASE was of interest because of the problem of erasing the line or lines at the right hierarchial level.

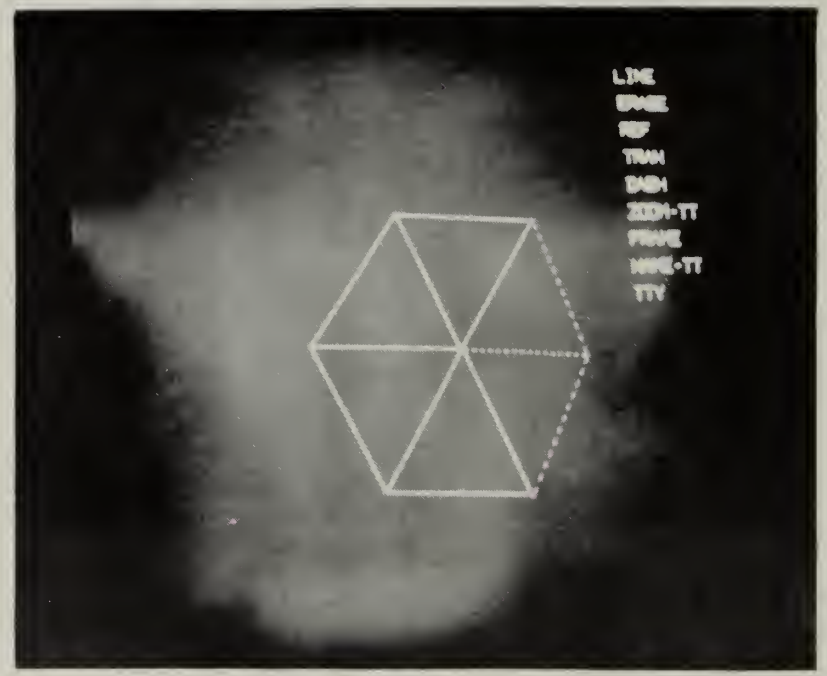
The manipulative functions implemented were REF (designating the reference point for the images or subpicture), TRAN (translation of the images or subpicture), ZOOM-TT (enlarge or dimish the image or subpicture) and DASH (change all lines in the image or subpicture to the dash mode). The REF function was chosen because it was needed in order to have an anchor point or reference point to manipulate the images around. The translation function was chosen as the main manipulative capability because it could be more easily implemented. It provided the same problems in respect to the hierarchial levels as the other manipulative functions. A function to increase and decrease the size of the images was desired; therefore, ZOOM-TT was selected as the best function to provide this capability. The problem of which hierarchial level should be changed to the dashed representation

was of sufficient interest to warrant the inclusion of the DASH function to the subset.

The storage and retrieval functions included in the subset are FRAME (forms and sequentially numbers subpictures) and NAME (forms and names images, and names subpictures). The function NAME is used to close out an image - group all the components (i.e., lines arcs and points) drawn since the previous image was closed out into one image - by placing a unique name in its header cell. FRAME closes out one subpicture and opens the next in a similar manner to the closing of the images by NAME. FRAME also automatically numbers the subpictures for future reference.

The function TTY was implemented. This gives the user the capability of entering point locations for LINE, REF, and TRANS by teletype.

The operating procedures utilized by a user are discussed below, from the standpoint of the actions required by a user and the responses that the system makes. The user loads the program, GPGSY, and executes it with the normal monitor commands. The nine functions appear on the CRT listed as a menu in the right margin. (See FIGURE 4.) The user selects the function LINE with a light pen pick and a cursor appears at the center of the screen. The user using the light pen guides the cursor to the desired position. When the cursor is in position, the user depresses function switch 1 (FNSW1), which stores the location in the display list as a move (vector with the beam blanked). Then the user guides the



DISPLAY DRAWN WITH
GPGSY

MENU FOR SELECTION

FIGURE 4

cursor to the next desired position and designates this point by again depressing FNSW1. This action places the selected location in the data list as a draw and a line segment appears with the two selected locations as the end points. The user continues drawing line segments as long as desired. The entering of line segments is only limited by the available core storage. When the user selects any of the other functions from the menu, the cursor disappears and some other interactive action takes place. Teletype messages giving the appropriate directions are used as responses so the user can utilize the system with hardly any prior instruction. After creating some object on the screen, the user can select any of the other eight functions.

The specific actions, which take place when a user selects a function by a light pen pick, are as follows:

(1.) ERASE

The teletype prints the following message, "SELECT IMAGE TO BE ERASED WITH LIGHT PEN." The user then takes a light pen pick on one of the lines to be erased and the entire image containing this line is erased from the screen.

(2.) REF

The cursor appears in the position of the present reference point. The user guides the cursor to the desired position and depresses FNSW1, which loads the new reference point into storage. All manipulative functions are now accomplished in respect to this new reference point.

(3.) TRAN

The cursor appears at the position of the present reference point. The user guides the cursor in the direction and the distance desired and then depresses FNSW1. The image or subpicture then translates in the direction and the distance that the cursor moved.

(4.) DASH

When the user selects this function, all the lines in the entire image or subpicture become dashed.

(5.) ZOOM-TT

The teletype prints the following message "INPUT UP TO 5 OCTAL DIGITS, NEGATIVE DIMINISHES." The user then inputs the incremental change in size that is desired. The function automatically limits the input from 0 (the image is shrunk to a point) to 37777 (the maximum size that the vector generator can scale a vector).

(6.) FRAME

The teletype prints the following message "SUBPIC_ CLOSED, SUBPIC_ OPENED" with the appropriate numbers in the blanks. No further action is required of the user.

(7.) NAME-TT

The teletype prints the following message "INPUT UP TO 5 CHARS." After the user enters the name of the image in five characters or less, the following message is printed by the teletype, "IMAGE CLOSED, NEW IMAGE OPENED."

(8.) TTY

The teletype prints the following message, "INPUT POINT (10 OCTAL DIGITS)." The user types in the coordinates of the point and in the cases of REF and TRAN no further action is required. With LINE, the teletype prints, "INPUT NEXT END POINT (10 OCTAL DIGITS) OR * TO END," and continues to accept points in this manner to draw a contiguous figure.

All the functions have appropriate messages which are typed by the teletype when the user commits an error, when all the images in a subpicture are used, or when all the subpictures are filled.

One of the original principles of the design of GPGL was to keep it as hardware independent as possible. GPGSY uses a CRT, teletypewriter, digital computer, light pen, one function switch, vector generator and character generator. Any computer graphics installation should have these devices (the character generator might be a software item), so the selected subset of GPGL can be considered relatively hardware independent. The hardware does certainly effect the implementation and in the case of GPGSY, the operating procedures. The vector generator develops new X and Y coordinates for the end points of the vectors which are to be drawn by the following formulas:

X'=DX+SC(X)

Y'=DY+SC(Y)

(X' is the new X coordinate, DX is a translation or offset increment, X is the old X coordinate and SC is the scale factor. Same for Y.)

The DPR2 has a hybrid array which automatically adds the DX and DY to every point in the display list. The register containing DX and DY is used in both TRAN and REF functions. The interesting point is that if an image is diminished by ZOOM-TT, the image is diminished, but the distance the image is from the reference point is also diminished because of the above formulas. If the user desires to diminish or magnify only the figure drawn, the user must move the reference point to the figure before using ZOOM-TT. Then ZOOM-TT is selected, the figure is diminished or magnified in position. This presents no serious problem to the user, but it does demonstrate the fact that the hardware (to be efficiently used) will dictate an order to execution for specific actions.

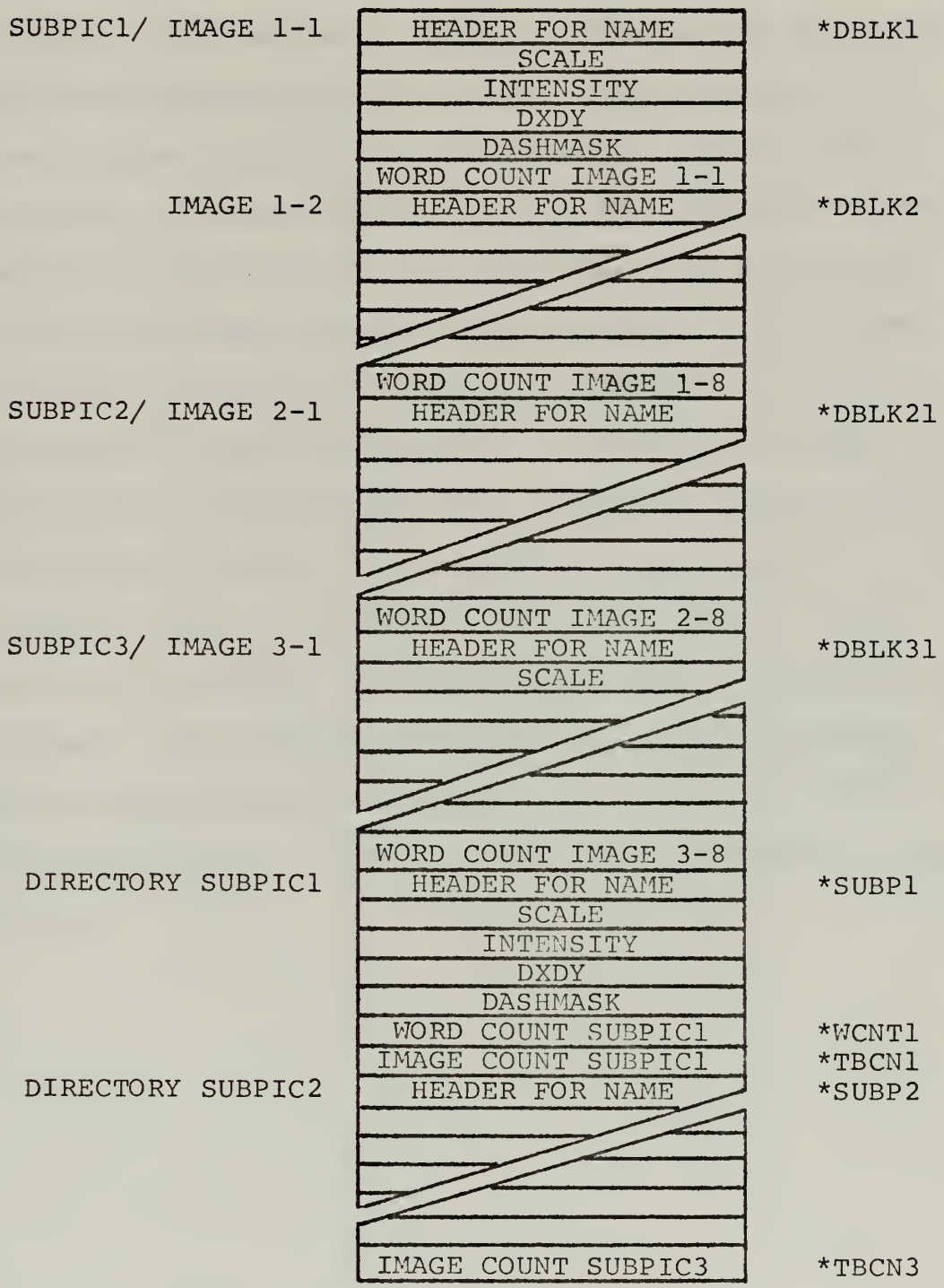
There was no problem implementing GPGSY in a conversational mode (a rapid computer response for each action of the user). Not all responses are graphical since many teletype messages are used as responses. This permits a user with little or no programming experience to use GPGSY, which is one of the design goals for GPGL.

The most interesting aspect in implementing GPGSY was the approach to a tri-level hierarchy within the components of the developed display. The lowest level, the image, is composed of any number of lines with the same scale, intensity and offset increments (DX and DY). Each image has a six cell directory which includes a header cell for the name, a cell for the scale, a cell for the intensity, a cell for DXDY, a cell for a dash mask (which is filled with an appropriate

mask when DASH is selected for the image) and a cell for the word count (the word count is the number of words in the display list and this number is loaded into the cell when the image is closed). Eight of these image directories make up a subpicture directory. The subpicture directory consists of seven cells which include the same six cells as the image directory plus a cell for the number of images filled in the frame (all eight images may not be used when the subpicture is closed). The system has three such directories; therefore, the picture can contain three subpictures. Thus, the directories form one large picture directory broken into three sequential subpicture directories; which in turn are broken into eight separate image directories. (See FIGURE 5.) The pointer for the directories is initiated pointing to the header of the first image in the first subpicture directory and is moved by computing an offset which is added to the pointer as the vector generator proceeds through the display list.

GPGSY provides up to 24 images contained in three subpictures, which make up one single picture. Any of the images can hold as many lines as the user desires up to the limit established by the available free core memory (4220 cells). This hierarchial level has cost the user using GPGSY a total of 165 memory cells. This storage loss could be reduced by about one third by loading scale, intensity, name and word count into half words and loading two words into one cell of the image directories. (Since

PICTURE DIRECTORY



*Variable names used in the computer program.

FIGURE 5

storage utilization was not a serious consideration and utilizing half words increases the complexity of the system this was not implemented in GPGSY.) Considering this savings, a tri-level hierarchy using the basic philosophy used in GPGL would cost the user approximately 110 memory cells per picture, or approximately 35 cells per subpicture added. If additional capabilities are required, such as rotation, this cost in storage would increase slightly. The three level hierarchy is quite adequate for demonstrating the capabilities of computer graphics to computer-oriented students and is adequate for most electrical engineering applications. Certain applications in the mechanical engineering field, especially those that pertain to gear trains and the movement of pistons and their related parts, may require more than the three levels provided by GPGSY. If these special applications are to be handled by GPGSY, the number of hierarchial levels of the system would have to be increased.

V. CONCLUSIONS

One of the considerations taken into account in selecting GPGL was the possibility of developing a single general purpose graphic language to handle all computer graphic applications. Although GPGL is a general purpose graphic language in that it can be used with many varied graphic applications, it is not suitable for all computer graphic applications. A mechanical engineering application, which would require more than three hierarchial levels, could not be effectively implemented with the present version of GPGL. Since many installations will use intelligent terminals with a small memory storage capacity, an all-inclusive general purpose language with its tremendous storage requirements could not be utilized by these installations. Most user's would use only some of the capabilities which would be provided by a single, all-inclusive general purpose language so that time and storage utilization would not be efficiently used. For maximum efficiency, a graphic language, which provided only the capabilities desired by the users, should be implemented. With the present state of the art, it is not feasible to design and create a single graphic language which can be used efficiently for all known applications of computer graphics.

In considering whether the subroutine or syntax approach should be used to implement a graphic language, it was previously mentioned that both approaches have advantages and

disadvantages. The decision as to which method should be used must be decided on an individual basis. Since the syntax method usually requires larger memory storage capacity and more programming expertise, it is felt that with the present state of the art, more installations will use the subroutine approach than the syntax approach.

The basic capabilities required of a general purpose graphic language can be provided with the following functions:

LINE
ARC
ERASE
TEXT
ROTATE
TRANSLATE
REFERENCE POINT
STORE
FETCH
NAME

GPGL provides all these functions plus many optional capabilities. The analysis capability and user-defined function capability are two of the more important options. The analysis capability is explicitly provided so that the user can do more than just draw pictures. The ability to program user-defined functions in GPGL gives the user the needed flexibility to handle many graphic processes which could not be as efficiently handled without this capability.

GPGSY, the implemented subset of GPGL, contains only five of the above required functions. The additional five functions should be added to GPGSY if the system is going to be used as an effective graphic system. These additional functions can be added without any great difficulty, but

the storage requirements would be significantly increased. In order to provide the user-defined function capability, the necessary teletype functions would have to be implemented. These include the user-defined instruction set, the teletype-system commands, the teletype-editor commands and the keyboard mode command. Implementing these functions would be a more difficult task than completing the basic requirement subset. The implementation of these functions and the user-defined functions, which would be created, would greatly increase the storage requirements of the system. In order to develop GPGSY into a true interactive, general purpose graphic system, these additional functions should be implemented, even though these changes are costly in storage and man hours.

The tri-level hierarchy of GPGSY provides a capability which is adequate for many applications. The cost of overhead can easily be reduced to 110 memory cells per picture (where three subpictures and 24 images are in a picture). The ease provided in manipulating the images and subpictures, which form the picture, is well worth this cost. Considering the trade-off between the necessary overhead in implementing hierarchial levels and the flexibility provided, the selection of three hierarchial levels appears to be excellent.

```
1.1         EXPUNGE
1.2         TITLE GPGSY
1.3         ENTRY GPGSY
1.4
1.5         [ EXTERNAL ENTRIES. $0FST:PRINTS THE STRING ON TELETYPE WHICH FOLLOWS THE CALL
1.6         [                   . $ICC:FETCHES A CHAR. INPUT BY TELETYPE (IN AR)
1.7         [                   . $WT1:HANDLES THE BACKGROUND,WHILE $ICC HANDLES FOREGROUND
1.10        [ ALL IN AMRMX (MONITOR)
1.11
1.12
1.13        [ INITIALIZATION
1.14        [ DBLK1. HEAD OF DIRECTORY
1.15        [ ROUTINE ENABLES SCOPE,AVG,FRAMECLOCK,FUNCTION SWITCHES
1.16
1.17        GPGSY:      0
1.20                    ARMD            SAVEAR      [SAVE THE AR REGISTER CONTENTS
1.21                    MDAR'F          FCLER
1.22                    ARMD            CLKPV
1.23                    MDAR'F          DBLK1
1.24                    ARMD            DBLK        [LOAD LOC OF HEAD OF DIRECTORY IN DBLK
1.25                    MD10'L
1.26                    60400JH
1.27                    MDIC'8'L
1.30                    00040JH
1.31                    MD11            SCAL        [SET SCALE
1.32                    MDD6            INTENS      [SET INTENSITY
1.33                    MD10'8'L
1.34                    1000JH
1.35
1.36
1.37        [ DRAW TEXT   ROUTINE
1.40
1.41        [ FFLAG=FLAG SET TO LOAD DTEXT WITH ENTRY ADDRESS
1.42        [ IDFLG=FLAG TO TELL CLOCK IMAGE IS DRAWN
1.43        [ CURFG=FLAG SET TO DRAW CURSOR
1.44        [ FLG1=FLAG USED TO KEEP FROM GETTING MULTIPLE HITS ON DESIGNATING POINTS
1.45        [ LPFLG=FLAG USED TO KEEP LIGHT PEN OFF 1 SEC.
1.46        [ LPCNT=CYCLE COUNTER TO KEEP LIGHT PEN OFF
1.47        [ INCTXT=INCREMENTS FETCH FOR LCG
1.50
1.51        [ ROUTINE DRAWS TEXT FOR FUNCTION ONE AT A TIME
1.52
1.53        DTEXT:      0
1.54                    MDAR            FFLAG
1.55                    JPLS            .+6
1.56                    MDAR            GPGSY
1.57                    MDAR'A          MASK5
1.60                    MDAR'8          MASK8
1.61                    ARMD            DTEXT
1.62                    ARMD'8          FFLAG
1.63                    ARX8'F
1.64                    ARMD            IDFLG
2.1         [DRAW TEXT ROUTINE CONT.
2.2
2.3                     MDAR            CURFG
2.4                     JPLS            .+2
2.5                     JUMP            DTEXT1
2.6                     S5AR'F
2.7                     JPAN            DFSPT       [ IF NEGATIVE JUMP TO DESIGNATE POINT
2.10                    ARX8'F
2.11                    ARMD            FLG1
2.12
2.13        DTEXT1:     0
2.14                    MDAR'F          LPLER
2.15                    ARMD            LPNPV
2.16                    MDAR            LPFLG
2.17                    JPLS            SKIP
2.20                    MDAR            LPCNT
2.21                    MDX8            LPMASK      [ MASK TO PERMIT LIGHT PEN TURN ON
2.22                    JPLS            SKIP1
2.23                    ARX8'F
2.24                    ARMD            LPCNT
2.25                    MDAR            ONE
2.26                    ARMD            LPFLG
2.27
2.30        SKIP:       MD10'8'L
2.31                    20JH                        [ TURN ON LIGHT PEN AGAIN
2.32        SKIP1:      MD10'A          MASK12      [ TURN OFF AVG
2.33                    MDD7            ZERO
2.34                    MDAR'F          DRTEXT
2.35                    MDAR'A          MASK5
2.36                    ARMD            INCTXT
2.37                    MDAR'F          DRTEXT
2.40                    ARMD            77735       [ DRTEXT IS TEXT DISPLAY LIST
2.41                    MDAR'F          TXLER
2.42                    ARMD            77736
2.43                    MDAR'F          TXLER
2.44                    ARMD            77737
2.45                    MDIC'A          CM14
2.46                    MDIC'8          TEN         [ DRAW TEXT
2.47                    MDAR            SAVEAR
2.50                    MDIR            DTEXT
```

```
3.1         [ DRAW VECTORS ROUTINE
3.2
3.3         [ FUNCTION FLAGS-IF SET GO TO ROUTINES FOR ACTION REQ
3.4         [ CURFG-FLAG SET TO DRAW CURSOR
3.5         [ BCNT-COUNT TO DETERMINE WHICH IMAGE IS TO BE DRAWN
3.6         [ TBCNT-TOTAL WHEN ALL IMAGES DRAWN
3.7         [ CONT-10,20,30 DEPENDING ON WHICH SUBPIC
3.10        [ FRFG1-FLAG TO SHOW IN SUBPIC2
3.11        [ FRFG2-FLAG TO SHOW IN SUBPIC3
3.12        [ IMCFG-FLAG ONCE SET DIRECTS THE PROGRAM FLOW THROUGH IMCLO
3.13
3.14        [ BASIC VECTOR DRAWING ROUTINE,JUMP OUT TO HANDLE FUNCTION ACTIONS        AND -
3.15        [ RETURN.LOADS NECCESSARY REGISTERS BY LOOPING IN TT FOR EACH IMAGE WHICH -
3.16        [ HAS SOMETHING IN IT TO DRAW.
3.17
3.20        ORVEC:      JUMP        .
3.21                    MDIC'A      CM14        [ TURN OFF LCG
3.22                    MDAR        FRAMEFG
3.23                    JSLS        FRAM1
3.24                    MDAR        TTYFG
3.25                    JSLS        TTY1
3.26                    MDAR        CURFG
3.27                    JSLS        PTRAC       [ CURSOR AND PEN TRACKING ROUTINE
3.30                    MDAR'F      ERLER
3.31                    ARMD        ERLPV
3.32                    MD10'0'L
3.33                    6D400JH                 [TURN ON AGAIN
3.34                    MDAR        TRAFG
3.35                    JSLS        TRA1
3.36                    MDAR        ZOOMFG
3.37                    JSLS        ZOOM1
3.40                    MDAR        DASHFG
3.41                    JSLS        DASH1
3.42                    MDAR        ERASEFG
3.43                    JSLS        ERASE1
3.44                    MDAR        NAMEFG
3.45                    JSLS        NAME1
3.46                    MDAR        REFG
3.47                    JSLS        REF1
3.50                    MDAR        DATA1
3.51                    MDX0        MZERO       [ NO ZEROS IN DISPLAY LIST JUMP SKIP2
3.52                    JPLS        .+2
3.53                    JUMP        SKIP2
3.54                    MDAR'L
3.55                    M005        DATA1-1
3.56                    ARMO        77756

4.1         [ DRAW VECTOR ROUTINE CONT.
4.2
4.3         TT:         ARX0'F                  [LOOP THAT LOADS REGISTERS AND DRAWS
4.4                     ARMD        ERLFG
4.5                     MDAR        BCNT
4.6                     MDX0        CONT
4.7                     JPLS        .+2
4.10                    JUMP        SKIP2
4.11                    MDAR        BCNT
4.12                    ARLS        2
4.13                    NOOP
4.14                    MDAE        BCNT
4.15                    MDAE        BCNT
4.16                    ARMD        TEMP4
4.17                    MDAR'F      DBLK1
4.20                    MDAE        TEMP4
4.21                    ARMD        TEMP4
4.22                    MDAR'I      TEMP4
4.23                    MDX0        MZERO
4.24                    JPLS        V1
4.25                    MDAR        FRFG2
4.26                    JPLS        SKIP2
4.27                    MDAR        FRAMEFG
4.30                    JSLS        FRAM3
4.31                    MDAR        FRFG1
4.32                    JPLS        TT
4.33                    JUMP        SKIP2
4.34        V1:         MDAR        IMCFG
4.35                    JPLS        IMCL1
4.36        V2:         MD11'I'X    TEMP4
4.37                    MDD6'I'X    TEMP4       [LOAD INTENSITY FOR THIS IMAGE
4.40                    MD07'I'X    TEMP4       [LOAD DXDY FOR THIS IMAGE
4.41                    MDAR'X      TEMP4
4.42                    MD1D'0'I    TEMP4       [IF DASHMASK IN SET DASH MODE
4.43        V3:         MDAR'X      BCNT
4.44                    MDIR'X      77756       [ DRAW VECTORS
4.45                    MDAR'N      ERLFG
4.46                    JPAN        .-1
4.47                    MD10'A      MASK14
4.50                    JUMP        TT
4.51        SKIP2:      MDAR        BCNT
4.52                    ARMD        TBCNT
4.53                    ARX0'F
4.54                    ARMO        BCNT
4.55                    ARMD        FRFG1
4.56                    ARMD        FRFG2
4.57                    MDAR        ONE
4.60                    ARMO        IOFLG
4.61                    MDAR        SAVAR
4.62                    MDIR        DRVEC
```

```
5.1         [ FRAME FUNCTION
5.2
5.3         [ FFG1=FLAG SET WHEN BEEN THROUGH ROUTINE ONCE AFTER FIRST LIGHT PEN HIT
5.4         [ FFG2=FLAG SET WHEN BEEN THROUGH ROUTINE ONCE AFTER SECOND LIGHT PEN HIT
5.5         [ WCONT=WORD COUNT CURRENT SUBPIC
5.6         [ TWCNT=TOTAL WORD COUNT OF SUBPIC BEING CLOSED AND ANY PREVIOUS SUBPICS
5.7         [ CONT1=COUNTER TO EXIT LOOP
5.10        [ ICFG=FLAG SET IN WBLOP WHICH SHOWS ALL IMAGES IN THE SUBPIC ARE CLOSED
5.11        [ TBCN1=COUNT OF IMAGES IN FIRST SUBPIC
5.12        [ TBCN2=COUNT OF SUBPICS + 10 IN SECOND SUBPIC
5.13
5.14        [ ROUTINE CLOSES OUT ONE SUBPIC AND OPENS THE NEXT ONE BY STORING THE WORD
5.15        [ COUNT OF THE OLD SUBPIC AND IMAGE COUNT THAT IS USED FOR THE OFFSET TO
5.16        [ THE DIRECTORY.
5.17        FRAM1:      JUMP        .
5.20                    ARX0'F
5.21                    ARMD        ICFG
5.22                    MDAR        FRAMEFG
5.23                    MDX0        ONE
5.24                    JPLS        FR1
5.25                    MDAR        FFG1
5.26                    JPLS        END2
5.27                    MDAR        ONE
5.30                    ARMD        FFG1
5.31                    JPSR        $OFST           [ PRINT VIA TELETYPE STRING
5.32        STRING '
5.33          SUBPIC 1 CLOSED, SUBPIC 2 OPENED
5.34        '
5.35        F1:         MDAR        WCONT           [ FETCH WORD COUNT OF CURRENT SUBPIC
5.36                    ARMD        WCNT1
5.37                    ARMD        TWCNT
5.40                    MDAR        TBCNT
5.41                    ARMD        TBCN1
5.42                    ARX0'F
5.43                    ARMD        WCONT
5.44                    MDAR        DBLK            [ADDRESS OF THE HEAD OF DIRECTORY
5.45                    MDAE        TWO
5.46                    ARMD        TEMP1
5.47                    MDAR        TWEN
5.50                    ARMD        CONT
5.51        OVER:       MDAR        ZERO            [ LOOP WHICH ZEROES INTENSITY
5.52                    ARMD'I      TEMP1
5.53                    MDAR'X      CONT1
5.54                    MDAR        TEMP1
5.55                    MDAE        SIX
5.56                    ARMD        TEMP1
5.57                    MDAR        CONT1
5.60                    MDX0        TEN
5.61                    JPLS        OVER
5.62                    ARX0'F
5.63                    ARMD        CONT1
5.64                    JUMP        END2
6.1         [ FRAME FUNCTION CONT.
6.2
6.3         FR1:        MDAR        FRAMEFG         [ THIS PORTION HANDLES CLOSING SUBPIC2 -
6.4                     MDX0        TWO             [ AND OPENING SUBPIC3.
6.5                     JPLS        FR2
6.6                     MDAR        FFG2
6.7                     JPLS        END2
6.10                    MDAR        ONE
6.11                    ARMD        FFG2
6.12                    JPSR        $OFST           [ PRINT STRING THAT FOLLOWS
6.13        STRING '
6.14          SUBPIC 2 CLOSED, SUBPIC 3 OPENED
6.15        '
6.16        F2:         MDAR        WCONT
6.17                    ARMD        WCNT2
6.20                    MDAE        TWCNT
6.21                    ARMD        TWCNT
6.22                    MDAR        TBCNT
6.23                    ARMD        TBCN2
6.24                    ARX0'F
6.25                    ARMD        WCONT
6.26                    MDAR'F      DBLK21
6.27                    MDAE        TWO
6.30                    ARMD        TEMP1
6.31                    MDAR        THIR
6.32                    ARMD        CONT
6.33                    JUMP        OVER
6.34
6.35        FR2:        JPSR        $OFST           [PRINT STRING THAT FOLLOWS
6.36        STRING '
6.37          ALL FRAMES FILLED
6.40        '
6.41                    MDAR'F      DATA1
6.42                    MDAE        TWCNT
6.43                    MDAE        WCONT
6.44                    MDAE        ONE
6.45                    ARMD        TEMP5
6.46                    MDAR        MASK9
6.47                    ARMD'I      TEMP5
6.50                    MDAR        TWCNT
6.51                    MDAE        TWO
6.52                    ARMD        TWCNT
6.53        END2:       MDIR        FRAM1
```

```
7.1         [ FRAME 3 FUNCTION
7.2
7.3         [ ROUTINE USED TO CHANGE BCNT(IMAGE COUNT) TO THE CORRECT FIGURE
7.4         [ IN ORDER TO COMPUTE THE CORRECT OFFSET TO THE DIRECTORY BECAUSE
7.5         [ ALL TEN IMAGES PER SUBPIC MIGHT NOT BE USED.
7.6
7.7         FRAM3:      JUMP        .
7.10                    MDAR        BCNT
7.11                    MDAE'N      TEN
7.12                    JPAN        .+2
7.13                    JUMP        FR5
7.14                    MDX0        MZERO
7.15                    JPLS        .+2
7.16                    JUMP        FR5
7.17                    MDAR        ONE
7.20                    ARMD        FRFG1
7.21                    MDAR        TEN
7.22                    ARMD        BCNT
7.23                    MDIR        FRAM3
7.24        FR5:        MDAR        FRAMEFG
7.25                    MDX0        TWO
7.26                    JPLS        FR6
7.27                    MDAR        TWEN
7.30                    ARMD        BCNT
7.31                    MDAR        ONE
7.32                    ARMD        FRFG1
7.33                    ARMD        FRFG2
7.34                    MDIR        FRAM3
7.35        FR6:        ARX0'F
7.36                    ARMD        FRFG1
7.37                    MDIR        FRAM3
10.1        [ NAME FUNCTION
10.2
10.3        [ TYPEFG=SET TO KEEP FROM PRINTING THE TELETYPE MESSAGE EACH ENTRY
10.4        [ NAMER=CELL WHERE NAME IS FORMED
10.5        [ CNTER=COUNTER TO CHECK ON NUMBER OF CHARS. ENTERED
10.6        [ WBADR=ADRESS OF HEADER OF IMAGE DIRECTORY OPEN
10.7        [ ICFG=FLAG SET IN WBLOP WHICH SHOWS ALL IMAGES IN SUBPIC ARE CLOSED
10.10
10.11       [ ROUTINE CALLS FOR NAME OF IMAGE TO BE INPUTTED,AND STORES IT
10.12       [ IN HEADER OF IMAGE DIRECTORY
10.13
10.14       NAME1:      JUMP        .
10.15                   MDAR        TYPEFG
10.16                   JPLS        AGAIN
10.17                   MDAR        ONE
10.20                   ARMD        TYPEFG
10.21                   JPSR        $OFST       [ PRINT INSTRUCTIONS BY TELETYPE
10.22       STRING '
10.23         INPUT UP TO 5 CHARS.
10.24       '
10.25       AGAIN:      MDAR'L
10.26                   JUMP        WAIT1       [ JUMP TO WAIT ROUTINE LOADED IN $AT1
10.27                   ARMD        $AT1
10.30                   JPSR        $ICC        [ FETCH TELETYPE CHAR.(IN AR ON RETURN)
10.31                   ARMD        TEMP1
10.32                   MDX0        FIFT
10.33                   JPLS        .+2
10.34                   JUMP        OVER1
10.35                   MDAR        CNTER
10.36                   MDX0        ZERO
10.37                   JPLS        N1
10.40                   MDAR        TEMP1
10.41                   ARLS        3D
10.42                   NOOP
10.43                   MDAR'O      NAMER
10.44                   ARMD        NAMER
10.45                   MDAR'X      CNTER
10.46                   JUMP        AGAIN
10.47       N1:         MDAR        CNTER
10.50                   MDX0        ONE
10.51                   JPLS        N2
10.52                   MDAR        TEMP1
10.53                   ARLS        22
10.54                   NOOP
10.55                   MDAR'O      NAMER
10.56                   ARMD        NAMER
10.57                   MDAR'X      CNTER
10.60                   JUMP        AGAIN
```

```
11.1        [ NAME FUNCTION CONT.
11.2
11.3        N2:             MDAR            CNTER
11.4                        MDAR            CNTER
11.5                        MDX@            TW@
11.6                        JPLS            N3
11.7                        MDAR            TEMP1
11.10                       ARLS            14
11.11                       N@@P
11.12                       MDAR'@          NAMER
11.13                       ARMD            NAMER
11.14                       MDAR'X          CNTER
11.15                       JUMP            AGAIN
11.16
11.17       N3:             MDAR            CNTER
11.20                       MDX@            THREE
11.21                       JPLS            N4
11.22                       MDAR            TEMP1
11.23                       ARLS            6
11.24                       N@@P
11.25                       MDAR'@          NAMER
11.26                       ARMD            NAMER
11.27                       MDAR'X          CNTER
11.30                       JUMP            AGAIN
11.31
11.32       N4:             MDAR            CNTER
11.33                       MDX@            FOUR
11.34                       JPLS            N5
11.35                       MDAR            TEMP1
11.36                       MDAR'@          NAMER
11.37                       ARMD            NAMER
11.40                       MDAR'X          CNTER
11.41                       JUMP            AGAIN
11.42       N5:             JPSR            $@FST           [ TOO MANY CHARS. ENTERED
11.43       STRING '
11.44         ONLY FIRST FIVE CHAR. ACCEPTED
11.45       '
11.46       OVER1:          JPSR            WBLOP
11.47                       MDAR            ICFG
11.50                       JPLS            IMCLR
11.51       N6:             MDAR            NAMER
11.52                       ARMD'I          WBADR
11.53                       MDAR            ICFG
11.54                       JPLS            FINIS
11.55                       MDAR            WBADR
11.56                       MDAL            FIVE
11.57                       ARMD            TEMP3
11.60                       MDAR            WCNT
11.61                       ARMD'I          TEMP3
11.62                       JUMP            FINIS
12.1        [ NAME FUNCTION CONT.
12.2
12.3        ERR1:           JPSR            $@FST           [ ERROR ALL IMAGES USED
12.4        STRING '
12.5          ALL IMAGES USED THIS RUBRIC
12.6        '
12.7        FINIS:          JPSR            $@FST           [ IMAGE CLOSED MSG. PRINTED
12.10       STRING '
12.11         IMAGE CLOSED,NEW IMAGE OPENED
12.12       '
12.13                       ARXH'F
12.14                       ARMD            NAMER
12.15                       ARMD            NAMEFG
12.16                       ARMD            CNTER
12.17                       ARMD            TYPEFG
12.20                       MDIR            NAME1
```

```
13•1        [ TELETYPE FUNCTION
13•2
13•3        [ PTCORD1=COORDINATES OF POINT ONE
13•4        [ PTCORD2=COORDINATES OF POINT TWO
13•5        [ PTCNT=POINT COUNTER USED TO GO TO CORRECT ENTRY
13•6        [ TTYCNT=TELETYPE INPUT COUNTER
13•7        [ LFG1=USED IN DESPT TO SHOW TELETYPE END POINT
13•10       [ TRANFG1=FLAG USED IN TRAN TO SHOW POINT RECEIVED
13•11       [ ERRFG= FLAG SET IN CHECK ROUTINE TO SHOW ERROR IN TTY
13•12
13•13       [ ROUTINE PERMITS TELETYPE ENTRY INSTEAD OF NORMAL LIGHT PEN ENTRY .
13•14       [ OF POINTS FOR LINE,TRAN,REF.SENDS LINE END POINTS TO DESPT TO .
13•15       [ LOAD INTO THE DISPLAY LIST AND LOAD IMAGE DIRECTORY.
13•16
13•17       TTY1:       JUMP        .
13•20                   JPSR        $OFST       [ PRINT INSTRUCTIONS BY TELETYPE
13•21       STRING '
13•22         INPUT POINT (10 OCTAL DIGITS)
13•23       '
13•24                   MDAR        REFG
13•25                   JPLS        RCOM
13•26                   MDAR        TRANFG
13•27                   JPLS        TRATY
13•30       AGAIN2:     MDAR'L
13•31                   JUMP        .WAIT1      [ LOAD JUMP TO WAIT ROUTINE IN INT1
13•32                   ARMD        $WT1
13•33                   JPSR        $ICC        [ FETCH TELETYPE CHAR.
13•34                   ARMD        TEMP1
13•35                   MDAR        LINEFG
13•36                   JPLS        .+2
13•37                   JUMP        .+5
13•40                   MDAR        TEMP1
13•41                   MDXO        FIVE
13•42                   JPLS        .+2
13•43                   JUMP        FINI1
13•44                   MDAR        TEMP1
13•45                   MDXO        FIFT
13•46                   JPLS        SKIP22
13•47                   MDAR        TTYCNT
13•50                   MDXO        TWELVE
13•51                   JPLS        .+2
13•52                   JUMP        .+2
13•53                   JUMP        ERR2
13•54                   ARXO'F
13•55                   ARMD        TTYCNT
13•56                   MDAR        REFG
13•57                   JPLS        RCOMP
13•60                   MDAR        TRANFG
13•61                   JPLS        TCOMP
13•62                   JUMP        COMP
13•63
14•1        [ TELETYPE FUNCTION CONT.
14•2
14•3        SKIP22:     JPSR        CHECK       [ CHECK CHAR.TO SEE IF DIGIT
14•4                    MDAR        ERRFG
14•5                    JPLS        ERR2        [ ERROR SO JUMP TO ERR2
14•6                    MDAR        TEMP1
14•7                    MDAR'A      SEVEN
14•10                   ARMD        TEMP1
14•11                   MDAR        PTCNT
14•12                   JPLS        SKIP23
14•13                   MDAR        TEMP1
14•14                   MDAR'O      PTCORD1
14•15                   ARMD        PTCORD1
14•16                   MDAR'X      TTYCNT
14•17                   MDXO        TWELVE
14•20                   JPLS        .+2
14•21                   JUMP        T1
14•22                   MDAR        PTCORD1
14•23                   ARLS        3
14•24                   ARMD        PTCORD1
14•25       T1:         JUMP        AGAIN2
14•26
14•27       SKIP23:     MDAR        TEMP1
14•30                   MDAR'O      PTCORD2
14•31                   ARMD        PTCORD2
14•32                   MDAR'X      TTYCNT
14•33                   MDXO        TWELVE
14•34                   JPLS        .+2
14•35                   JUMP        T2
14•36                   MDAR        PTCORD2
14•37                   ARLS        3
14•40                   ARMD        PTCORD2
14•41       T2:         JUMP        AGAIN2
14•42
14•43       COMP:       MDAR        PTCNT
14•44                   JPLS        COMP1
14•45                   MDAR'X      PTCNT
14•46       T3:         JPSR        $OFST
14•47       STRING '
14•50         INPUT NEXT END POINT(10 OCTAL DIGITS) OR . TO END
14•51       '
14•52                   ARXO'F
14•53                   ARMD        PTCORD2
14•54                   JUMP        AGAIN2
14•55
14•56       COMP1:      MDAR        ONE
14•57                   ARMD        LFG1
14•60                   MDAR'X      PTCNT
14•61                   JUMP        JU
14•62
```

```
15.1        [ TELETYPE FUNCTIN CONT.
15.2
15.3        FINI1:      ARX8'F
15.4                    ARMD        LINEFG
15.5                    ARMD        TTYFG
15.6                    ARMD        PTC8RD1
15.7                    ARMD        PTC8RD2
15.10                   ARMD        PTCNT
15.11                   MDIR        TTY1
15.12
15.13       RC8M:       MDAR        8NE         [ THIS P8RTI8N HANDLES REF FUNCTI8N
15.14                   ARMD        PTCNT
15.15                   JUMP        AGAIN2
15.16       RC8MP:      MDAR        PTC8RD2
15.17                   MDAR'A      MASK6
15.20                   ARMD        NUMB2
15.21                   ARX8'F
15.22                   ARMD        REFG1
15.23                   JUMP        FINI1
15.24
15.25       TRATY:      MDAR        8NE         [ THIS P8RTI8N HANDLES TRAN FSNCTI8N
15.26                   ARMD        PTCNT
15.27                   JUMP        AGAIN2
15.30
15.31       TC8MP:      MDAR        PTC8RD2
15.32                   MDAR'A      MASK6
15.33                   ARMD        NUMB2
15.34                   ARX8'F
15.35                   ARMD        TRANFG1
15.36                   JUMP        FINI1
15.37
15.40       ERR2:       MDAR        LINEFG      [ ERR8R HANDLING PART
15.41                   JPLS        .+2
15.42                   JUMP        ERR3
15.43                   JPSR        $8FST       [ PRINT STRING F8R LINE ERR8R
15.44       STRING '
15.45         ERR8R,INPUT FIRST END P8INT AGAIN (10 8CTAL DIGITS) AND C/R
15.46       '
15.47                   ARX8'F
15.50                   ARMD        PTCNT
15.51                   JUMP        ERR4
15.52
15.53       ERR3:       JPSR        $8FST       [ PRINT STRING F8R TRAN AND REF
15.54       STRING '
15.55         ERR8R,INPUT P8INT AGAIN (10 8CTAL DIGITS) AND C/R
15.56       '
15.57                   MDAR        8NE
15.60                   ARMD        PTCNT
15.61       ERR4:       ARX8'F
15.62                   ARMD        ERRFG
15.63                   ARMD        TTYCNT
15.64                   JUMP        AGAIN2
16.1        [ ERASE FUNCTI8N
16.2
16.3        [ R8UTINE SETS LIGHT PEN PIV8T F8R ERASE,TURNS LIGHT PEN 8N          ,*
16.4        [ AND GIVES APPR8PRIATE INSTRUCTI8NS T8 USER.
16.5
16.6        ERASE1:     JUMP        .
16.7                    MDAR'F      ELPLER
16.10                   ARMD        LPNPV
16.11                   MDIC'8'L
16.12                   20/H
16.13                   MDAR        TYPEFG
16.14                   JPLS        FINI2
16.15                   JPSR        $8FST
16.16       STRING '
16.17         SELECT IMAGE T8 ERASE WITH LIGHT PEN
16.20       '
16.21                   MDAR        8NE
16.22                   ARMD        TYPEFG
16.23       FINI2:      MDIR        ERASE1
16.24
16.25
16.26       [ ERASE LIGHT PEN HANDLER
16.27
16.30       [ BCNT=C8UNTR WHICH SH8WS WHAT IMAGE IS CURRENTLY BEING DRAWN
16.31       [ TACNT=T8TAL W8RD C8UNT IN DISPLAY LIST WHEN LAST SUBPIC CL8SED
16.32       [ DATA1=HEADER F8R DISPLAY LIST
16.33       [ D8LK=ADDRESS 8F HEADER F8R DIRECT8RY
16.34
16.35       [ R8UTINE L8CATES WHICH IMAGE IS BEING REFRESHED AT TIME 8F PICK -
16.36       [ BY BCNT AND ERASES THE ENTIRE IMAGE
16.37
16.40       ELPLER:     0
16.41                   ARMD        SAR1
16.42                   MDAR        DATA1
16.43                   MDX8        MZER8
16.44                   JPLS        E1
16.45                   JUMP        END3
16.46       E1:         MDAR        BCNT
16.47                   MDX8        8NE
16.50                   JPLS        E2
16.51                   MDAR'F      DATA1
16.52                   ARMD        TEMP5
16.53                   JUMP        AGAIN3
16.54
16.55       [ ERASE LIHT PEN HANDLER
16.56
16.57       E2:         MDAR        BCNT
16.60                   ARLS        2
16.61                   N88P
16.62                   MDAE        BCNT
16.63                   MDAE        BCNT
16.64                   MDAE'N      SIX
16.65                   ARMD        TEMP1
```

```
17.1        [ ERASE LIGHT PEN HANDLER CONT.
17.2
17.3                    MDAR            DBLK
17.4                    MDAE            TEMP1
17.5                    MDAE'N          BNE
17.6                    ARMD            TEMP1
17.7                    MDAR'1          TEMP1
17.10                   ARMD            TEMP4
17.11                   MDAR'X          TEMP1
17.12                   MDAR'F          DBLK21
17.13                   MDX8            TEMP1
17.14                   JPLS            .+4
17.15                   MDAR            TWCNT
17.16                   ARMD            TEMP4
17.17                   JUMP            E3
17.20                   MDAR'F          DBLK31
17.21                   MDX8            TEMP1
17.22                   JPLS            .+4
17.23                   MDAR            TWCNT
17.24                   ARMD            TEMP4
17.25                   JUMP            E3
17.26                   MDAR            FRAMEFG
17.27                   JPLS            .+2
17.30                   JUMP            .+5
17.31                   MDAR            TEMP4
17.32                   MDAE            TWCNT
17.33                   ARMD            TEMP4
17.34                   JUMP            E3
17.35                   MDAR            TEMP4
17.36                   ARMD            TEMP4
17.37       E3:         MDAR'F          DATA1
17.40                   MDAE            TEMP4
17.41                   ARMD            TEMP5
17.42
17.43       AGAIN3:     MDAR'I'M        TEMP5
17.44                   MDAR'A          BNE
17.45                   JPLS            E4
17.46                   MDAR'I          TEMP5
17.47                   MDAR'A          MASK10
17.50                   ARMD'I          TEMP5
17.51                   MDAR'X          TEMP5
17.52                   JUMP            AGAIN3
17.53       E4:         MDAR'I          TEMP5
17.54                   MDAR'A          MASK10
17.55                   ARMD'I          TEMP5
17.56                   MD10'A          MASK2
17.57       END3:       ARX8'F
17.60                   ARMD            TYPEFG
17.61                   ARMD            ERASEFG
17.62                   MDAR            SAR1
17.63                   JUMP'I          ELPLER
20.1        [REFERENCE POINT FUNCTION
20.2
20.3        [ NUMB1-NUMBER TO SUBTRACT TO BE SUBTRACTED IN SUB2
20.4        [ NUMB2-NUMBER WHICH NUMB1 IS SUBTRACTED FROM IN SUB2
20.5        [ DXDY-LOC WHERE DIFFERENCE IN X (0-14 BITS) AND Y (15-29) IS STORED
20.6        [ MOVFG- SET IF DISPLAY LIST WORD WAS A MOVE
20.7        [ REFPT- CELL IN WHICH REFERENCE POINT IS STORED
20.10       [ WBADR-ADRESS OF HEADER OF OPEN IMAGE
20.11
20.12       [ ROUTINE LOADS SUBTRACTS OLD REFERENCE POINT FROM THE NEW ONE DESIGNATED -
20.13       [ AND LOADS RESULT IN IMAGE DIRECTORY AS OFFSET INCREMENT(TO BE LOADED IN -
20.14       [ D7 REGISTER) AND SUBTRACTS SAME RESULT FROM EACH WORD IN IMAGE DISPLAY LIST.
20.15       [ LOADS NEW REFERENCE POINT IN REFPT
20.16
20.17       REF1:           JUMP            .
20.20                       MDAR            TTYFG
20.21                       JSLS            TTY1
20.22       RCOMP1:         MDAR            REFG1
20.23                       JPLS            END
20.24                       MDAR            ICFG
20.25                       JPLS            IMCLR
20.26                       MDAR            REFPT
20.27                       ARMD            NUMB1
20.30                       JPSR            SUB2
20.31                       MDAR            NUMB1
20.32                       ARMD            DXDY
20.33                       MDAR            NUMB2
20.34                       ARMD            REFPT
20.35                       JPSR            WBLOP           [ JUMP TO FIND OUT OPEN IMAGE
20.36                       MDAR            WBADR
20.37                       MDAE'N          BNE
20.40                       ARMD            TEMP1
20.41                       MDAE'N          DBLK            [ SUBTRACT TO HEAD OF DIRECTORY ADDRESS
20.42                       JPAN            .+2
20.43                       JUMP            .+4
20.44                       ARX8'F
20.45                       ARMD            TEMP2
20.46                       JUMP            RO
20.47                       MDAR'I          TEMP1
20.50                       ARMD            TEMP2
20.51                       MDAR            FRAMEFG
20.52                       JPLS            .+2
20.53                       JUMP            RO
20.54                       MDAR'X          TEMP1
20.55                       MDAR'F          DBLK21
20.56                       MDX8            TEMP1
20.57                       JPLS            .+2
20.60                       JUMP            .+3
20.61                       MDAR'F          DBLK31
20.62                       MDX8            TEMP1
20.63                       JPLS            .+4
```

```
21.1        [ REFERENCE POINT FUNCTION CONT.
21.2
21.3                    MDAR            TWCNT
21.4                    ARMD            TEMP2
21.5                    JUMP            RD
21.6                    MDAR            TEMP2
21.7                    MDAE            TWCNT
21.10                   ARMD            TEMP2
21.11
21.12       R0:         MDAR'F          DATA1
21.13                   MDAE            TEMP2
21.14                   ARMD            TEMP2
21.15       R1:         MDAR            DXDY
21.16                   ARMD            NUMB1
21.17                   MDAR'I'M        TEMP2
21.20                   MDAR'A          BNE
21.21                   JPLS            R2
21.22                   MDAR'I          TEMP2
21.23                   MDAR'A          BNE
21.24                   JPLS            .+3
21.25                   MDAR            BNE             [WORD IN DISPLAY LIST IS MOVE,SET MOVFG
21.26                   ARMD            MOVFG
21.27                   MDAR'I          TEMP2
21.30                   MDAR'A          MASK6
21.31                   ARMD            NUMB2
21.32                   JPSR            SUB2
21.33                   MDAR            MOVFG
21.34                   MDXB            BNE
21.35                   JPLS            .+3
21.36                   MDAR            NUMB1
21.37                   JUMP            .+3
21.40                   MDAR            NUMB1
21.41                   MDAR'B          BNE
21.42                   MDAR'A          MASK7           [ MASK7=7777677777
21.43                   ARMD'I          TEMP2
21.44                   ARXB'F
21.45                   ARMD            MOVFG
21.46                   MDAR'X          TEMP2
21.47                   JUMP            R1
21.50       R2:         MDAR            DXDY            [ THIS PORTION HANDLES WORD WITH EOL =
21.51                   ARMD            NUMB1           [ BIT SET(LAST WORD CURRENTLY IN IMAGE)
21.52                   MDAR'I          TEMP2
21.53                   MDAR'A          MASK6
21.54                   ARMD            NUMB2
21.55                   JPSR            SUB2
21.56                   MDAR            NUMB1
21.57                   MDAR'B          CIM1
21.60                   ARMD'I          TEMP2
21.61                   MDAR            ARADR
21.62                   MDAE            THREE
21.63                   ARMD            TEMP1
21.64                   MDAR            REFPT
21.65                   ARMD'I          TEMP1

22.1        [ REFERENCE POINT CONT.
22.2
22.3        R3:         ARXB'F
22.4                    ARMD            REFG
22.5                    ARMD            TTYFG
22.6                    MDAR            BNE
22.7                    ARMD            REFG1
22.10       END:        MDIR            REF1
22.11
22.12
22.13       [ CHECK FOR DIGIT IN CODE
22.14
22.15       [ ROUTINE CHECKS TELETYPE INPUT AND MAKES SURE IT IS A DIGIT
22.16       [ SETS ERRFG IF AN ERROR IS DETECTED
22.17
22.20       CHECK:      JUMP            .
22.21                   MDAR            TEMP1
22.22                   MDAR'A'L
22.23                   77770
22.24                   MDXB            TAEX
22.25                   JPLS            .+2
22.26                   MDIR            CHECK
22.27                   MDAR            BNE
22.30                   ARMD            ERRFG
22.31                   MDIR            CHECK
```

```
23.1        [ TRANSLATION FUNCTION
23.2
23.3        [ TRANFG1=FLAG WHEN ZERO SHOWS A POINT HAS BEEN DESIGNATED
23.4        [ NUMB1=NUMBER TO SUBTRACT IN SUB2
23.5        [ NUMBE2=NUMBER TO BE SUBTRACTED FROM IN SUB2
23.6        [ WBADR=HEADER OF IMAGE DIRECTORY OPEN
23.7
23.10       [ ROUTINE HAS SUB2 SUBTRACT OLD REFERENCE POINT FROM POINT DESIGNATED -
23.11       [ AS TRANSLATION DIRECTION AND DISTANCE.PROGRAM ENTERS EVERY CYCLE AFTER
23.12       [ TRAN HIT SO IF POINTS NOT DESIGNATED YET,JUMP TO END1.
23.13
23.14       TRAN1:      JUMP        .
23.15                   MDAR        TTYFG
23.16                   JSLS        TTY1
23.17       TRAN2:      MDAR        TRANFG1
23.20                   JPLS        END1
23.21                   MDAR        REFPT
23.22                   ARMD        NUMB1
23.23                   JPSR        SUB2
23.24                   JPSR        WBLOP
23.25                   MDAR        ICFG
23.26                   JPLS        IMCLO
23.27       TRAN3:      MDAR        WBADR
23.30                   MDAE        THREE
23.31                   ARMD        TEMP2
23.32                   MDAR        NUMB1
23.33                   ARMD'I      TEMP2
23.34                   ARXO'F
23.35                   ARMD        TRANFG
23.36                   MDAR        ONE
23.37                   ARMD        TRANFG1
23.40       END1:       MDIR        TRAN1
23.41
23.42

24.1        [SUBTRACT TWO POINT ROUTINE
24.2
24.3        [ NUMB1=NUMBER TO SUBTRACT
24.4        [ NUMB2=NUMBER SUBTRACTED FROM
24.5
24.6        [ ROUTINE SUBTRACTS TWO POINTS,HALF WORD AT A TIME,UPPER HALF IS X -
24.7        [ COORD. AND LOWER HALF THE Y COORD.
24.10
24.11       SUB2:       JUMP        .
24.12                   MDAR        NUMB1
24.13                   MDAR'A      MASK3           [ MASK3=0000077776
24.14                   ARMD        TEMP3           [STORE RP
24.15                   MDAR        NUMB2
24.16                   MDAR'A      MASK3           [AND MASK3 WITH LP1
24.17                   ARMD        TEMP4           [STORE IN TEMP4
24.20                   MDAE'N      TEMP3
24.21                   ARMD        TEMP3
24.22                   MDXO        MZERO
24.23                   JPLS        .+3
24.24                   ARXO'F
24.25                   ARMD        TEMP3
24.26                   MDAR        TEMP3
24.27                   MDAR'A      MASK3
24.30                   ARMD        TEMP3
24.31                   MDAR        NUMB1
24.32                   ARRS        17              [RIGHT SHIFT 15
24.33                   NOOP
24.34                   ARMD        TEMP4           [STORE RP (0-14) IN TEMP (15-29)
24.35                   MDAR        NUMB2
24.36                   ARRS        17
24.37                   NOOP
24.40                   ARMD        TEMP5
24.41                   MDAE'N      TEMP4
24.42                   ARMD        TEMP4
24.43                   MDXO        MZERO
24.44                   JPLS        .+3
24.45                   ARXO'F
24.46                   ARMD        TEMP4
24.47                   MDAR        TEMP4
24.50                   ARLS        17
24.51                   NOOP
24.52                   MDAR'A      MASK4           [ MASK4=7777700000
24.53                   MDAR'O      TEMP3           [ OR DX (0-14) AND DY (15-29)
24.54                   ARMD        NUMB1
24.55                   MDXO        MZERO
24.56                   JPLS        .+3
24.57                   ARXO'F
24.60                   ARMD        NUMB1
24.61                   MDIR        SUB2
24.62
24.63
```

```
25.1
25.2
25.3        [ ZCNTER=COUNTER TO COUNT INPUT DIGITS
25.4        [ FAC1=CELL WHERE INPUT NUMBER FORMED
25.5        [ ERRFG=FLAG SET IN CHECK IF ERROR
25.6        [ WBADR=ADRESS OF HEADER OF DIRECTORY OF IMAGE OPEN
25.7
25.10       [ ROUTINE ACCEPTS TELETYPE DIGITS TO SET SCALE IN AVG FROM 0-1
25.11       [ SENDS INPUTTED CHARS. TO CHECK FOR VERIFICATION,IF ERROR USER -
25.12       [ MUST INPUT AGAIN
25.13
25.14       Z88M1:          JJMP            .
25.15                       JPSR            $OFST
25.16       STRING '
25.17         INPUT UP TO 5 OCTAL DIGITS,NEGATIVE DIMINISHES
25.20       '
25.21       AGAIN1:         MDAR'L
25.22                       JJMP            WAIT1
25.23                       ARMD            $WT1
25.24                       JPSR            $ICC
25.25                       ARMD            TEMP1
25.26                       MDX8            FIFT
25.27                       JPLS            Z1
25.30                       MDAR            FIVE
25.31                       MDAE'N          ZCNTER
25.32                       JPAN            Z88M2
25.33                       MDAR            FAC1
25.34                       ARRS            3
25.35                       JJMP            Z88M2
25.36       Z1:             MDAR            ZCNTER
25.37                       MDX8            FIVE
25.40                       JPLS            .+2
25.41                       JJMP            ERROR
25.42                       JPSR            CHECK
25.43                       MDAR            ERRFG
25.44                       JPLS            ERROR           [IF ERROR FLAG SET JUMP TO ERROR
25.45                       MDAR            TEMP1
25.46                       MDAR'A          SEVEN
25.47                       MDAR'O          FAC1
25.50                       ARMD            FAC1
25.51                       MDAR'X          ZCNTER
25.52                       MDX8            FIVE
25.53                       JPLS            .+2
25.54                       JJMP            AGAIN1
25.55                       MDAR            FAC1
25.56                       ARLS            3
25.57                       ARMD            FAC1
25.60                       JJMP            AGAIN1
25.61
26.1        [ Z88M FUNCTION CONT.
26.2
26.3        Z88M2:          MDAR            FAC1
26.4                        ARLS            17
26.5                        NOOP
26.6                        ARMD            FAC1
26.7                        JPSR            WBLOP
26.10                       MDAR            ICFG
26.11                       JPLS            IMCLO
26.12       Z2:             MDAR            WBADR
26.13                       MDAE            ONE
26.14                       ARMD            TEMP3
26.15                       MDAR'I          TEMP3
26.16                       MDAE            FAC1
26.17                       ARMD            FAC1
26.20                       MDAR            ICFG
26.21                       JPLS            Z4
26.22                       MDAR            MXSCL
26.23                       MDAE'N          FAC1
26.24                       JPAN            .+2
26.25                       JJMP            Z3
26.26                       MDAR            MXSCL
26.27                       ARMD'I          TEMP3
26.30                       JJMP            Z5
26.31       Z3:             MDAR            FAC1
26.32                       JPAN            .+2
26.33                       JJMP            Z4
26.34                       MDAR            ZERO
26.35                       ARMD'I          TEMP3
26.36                       JJMP            Z5
26.37       Z4:             MDAR            FAC1
26.40                       ARMD'I          TEMP3
26.41       Z5:             ARX8'F
26.42                       ARMD            ZCNTER
26.43                       ARMD            FAC1
26.44                       ARMD            Z88MFG
26.45                       MDIR            Z88M1
26.46
26.47       ERROR:          JPSR            $OFST           [ ERROR SO PRINT FOLLOWING STRING
26.50       STRING '
26.51         INPUT UP TO 5 OCTAL DIGITS (0-37777) AND C/R
26.52       '
26.53                       ARX8'F
26.54                       ARMD            ERRFG
26.55                       ARMD            ZCNTER
26.56                       ARMD            FAC1
26.57                       JJMP            AGAIN1
26.60
26.61       WAIT1:                          0               [ DO NOTHING LOOP WAITING FOR $ICC -
26.62                       JJMP            .-1             [ TO FETCH A TELETYPE CHAR.
```

```
27.1        [ DASH FUNCTION
27.2
27.3        [ ABADR=ADDRESS OF HEAD OF DIRECTORY OF IMAGE OPEN
27.4
27.5        [ ROUTINE LOADS DASHMASK IN IMAGE DIRECTORY OF OPEN IMAGE AND RESETS DASHFG
27.6
27.7        DASH1:      JUMP        .
27.10                   JPSR        ABLOP
27.11                   MDAR        ICFG
27.12                   JPLS        IMCL0
27.13       D1:         MDAR        ABADR
27.14                   MDAE        FOUR
27.15                   ARMD        TEMP1
27.16                   MDAR        DASHMASK
27.17                   ARMD'I      TEMP1
27.20                   ARX8'F
27.21                   ARMD        DASHFG
27.22       FINI3:      MDIR        DASH1
27.23
27.24       [ IMAGE CLOSED ROUTINE
27.25
27.26       [ ICFG=FLAG WHICH IS SET IN WBLOP WHICH SHOWS ALL IMAGES ARE CLOSED
27.27       [ IMCFG=FLAG ONCE SET DIRECTS PROGRAM FLOW THROUGH IMCL0
27.30       [ ICFG1=FLAG THAT SHOWS SUBPIC1 IS MANIPULATED
27.31       [ ICFG2=FLAG THAT SHOWS SUBPIC2 IS MANIPULATED
27.32       [ ICFG3=FLAG THAT SHOWS SUBPIC3 IS MANIPULATED
27.33       [ SUBP1,2,3= HEADS OF SUBPIC DIRECTORIES
27.34
27.35       [ ROUTINE CAUSES MANIPULATORS TO OPERATE ON ENTIRE SUBPIC NOT JUST ONE IMAGE
27.36       [ IMCL0 PORTION HANDLES NECESSARY ACTION ON SUBPICS FOR NAME,DASH,ZOOM AND TRAN
27.37
27.40       IMCL0:      MDAR        ONE
27.41                   ARMD        IMCFG
27.42                   MDAR        TBCNT
27.43                   MDAE'N      SEVEN
27.44                   JPAN        .+4
27.45                   MDAR        TBCNT
27.46                   MDX8        TEN
27.47                   JPLS        IM1
27.50                   MDAR        ONE
27.51                   ARMD        ICFG1
27.52                   MDAR'F      SUBP1
27.53                   ARMD        IMC
27.54                   JUMP        IM1C
27.55       IM1:        MDAR        TBCNT
27.56                   MDAE'N      SEVTEEN
27.57                   JPAN        .+4
27.60                   MDAR        TBCNT
27.61                   MDX8        TWEN
27.62                   JPLS        IM2
27.63                   MDAR        ONE
27.64                   ARMD        ICFG2
27.65                   MDAR'F      SUBP2

30.1        [ IMAGE CLOSED ROUTINE CONT.
30.2
30.3                    ARMD        IMC
30.4                    JUMP        IM1C
30.5        IM2:        MDAR'F      SUBP3
30.6                    ARMD        IMC
30.7                    MDAR        ONE
30.10                   ARMD        ICFG3
30.11
30.12       IM1C:       MDAR        IMC
30.13                   ARMD        ABADR
30.14                   MDAR        NAMEFG
30.15                   JPLS        .+2
30.16                   JUMP        IM11
30.17                   JUMP        N6
30.20       IM11:       MDAR        DASHFG
30.21                   JPLS        .+2
30.22                   JUMP        IM12
30.23                   JUMP        D1
30.24       IM12:       MDAR        ZOOMFG
30.25                   JPLS        .+2
30.26                   JUMP        IM13
30.27                   JUMP        Z2
30.30       IM13:       MDAR        TRANFG
30.31                   JPLS        .+2
30.32                   JUMP        IM14
30.33                   JUMP        TRAN3
30.34       IM14:       MDAR'I      IMC
30.35                   ARMD        NUMB1
30.36                   JPSR        SUB2
30.37                   MDAR        NUMB1
30.40                   ARMD'I      IMC
30.41                   JUMP        R3
30.42
30.43       [ IMCL1 PORTION HANDLES THE ACTIONS REQUIRED FOR OPERATING
30.44       [ ON THE WHOLE SUBPIC FOR DRVEC
30.45
30.46       IMCL1:      MDAR        ICFG1
30.47                   JPLS        .+2
30.50                   JUMP        IMC1
30.51                   MDAR        BCNT
30.52                   MDAE'N      SEVEN
30.53                   JPAN        .+2
30.54                   JUMP        IMC1
30.55                   MDAR'F      SUBP1
30.56                   ARMD        IMC
30.57                   JUMP        IMC1C
30.60       IMC1:       MDAR        ICFG2
30.61                   JPLS        .+2
30.62                   JUMP        IMC2
30.63                   MDAR        BCNT
30.64                   MDAE'N      SEVEN
30.65                   JPAN        V2
```

```
31.1        [ IMAGE CLOSED ROUTINE CONT.
31.2
31.3                    MDAR            BCNT
31.4                    MDAE'N          SEVTEEN
31.5                    JPAN            .+2
31.6                    JUMP            IMC2
31.7                    MDAR'F          SUBP2
31.10                   ARMD            IMC
31.11                   JUMP            IMC10
31.12       IMC2:       MDAR            ICFG3
31.13                   JPLS            .+2
31.14                   JUMP            V2
31.15                   MDAR            BCNT
31.16                   MDAE'N          SEVTEEN
31.17                   JPAN            V2
31.20                   MDAR'F          SUBP3
31.21                   ARMD            IMC
31.22       IMC10:      MDAR'I'X        TEMP4
31.23                   ARRS            17
31.24                   N99P
31.25                   ARMD            TEMP6
31.26                   MDAE'I'X        IMC
31.27                   ARRS            17
31.30                   N99P
31.31                   ARMD            TEMP1
31.32                   MDAR            TEMP6
31.33                   MDAE            TEMP1
31.34                   ARMD            TEMP1
31.35                   JPAN            .+10
31.36                   MDAE'N          MXSCL
31.37                   JPAN            .+6
31.40                   MDAR            MXSC1
31.41                   ARMD            TEMP1
31.42                   JUMP            .+3
31.43                   ARX8'F
31.44                   ARMD            TEMP1
31.45                   MDAR            TEMP1
31.46                   ARLS            17
31.47                   N99P
31.50                   ARMD            TEMP1
31.51                   MD11            TEMP1
31.52                   MDD6'I'X        TEMP4
31.53                   MDAR'X          IMC
31.54                   MDAR'I'X        IMC
31.55                   ARAR'N
31.56                   ARMD            NUMB1
31.57                   MDAR'I'X        TEMP4
31.60                   ARMD            NUMB2
31.61                   JPSR            SUB2
31.62                   MDD7            NUMB1
31.63                   MDAR'I'X        IMC
31.64                   ARMD            TEMP1
31.65                   MD10'8          TEMP1
32.1        [ IMAGE CLOSED ROUTINE CONT.
32.2
32.3                    JUMP            V3
32.4
32.5        [ LIGHT PEN HANDLER FOR TEXT
32.6
32.7        [ TXCNT=COUNTER FOR WHICH TEXT LINE WAS BEING DRAWN WHEN LIGHT PEN HIT
32.10       [ COUNT
32.11       [ COUNT=COUNTER FOR NUMBER OF POINTS IN DESPT
32.12       [ TRCRD=TRACK COORDINATES OF CURSOR
32.13       [ CURFG=FLAG SET TO DRAW CURSOR
32.14       [ DATA1=HEADER FOR DISPLAY LIST
32.15
32.16       [ ROUTINE HANDLES LIGHT PEN HITS ON THE TEXT LINES OF THE MENU,COUNTER -
32.17       [ IS INCREMENTED IN END OF TEXT HANDLER. COUNT DETERMINES WHICH FUNCTION -
32.20       [ PICKED AND THEN APPROPRIATE FLAGS SET.
32.21
32.22       LPLER:      0
32.23                   ARMD            LPSAV
32.24                   MD10'A          MASK2
32.25                   MDAR            TXCNT1
32.26                   ARMD            TXCNT
32.27                   ARX8'F
32.30                   ARMD            LPFLG
32.31                   MDAR            TXCNT
32.32                   MDX8            ONE
32.33                   JPLS            J1
32.34                   MDAR            ONE
32.35                   ARMD            LINEFG
32.36                   ARMD            CURFG
32.37                   ARX8'F
32.40                   ARMD            COUNT
32.41                   JUMP            JP
32.42
```

```
33.1        ( TEXT LIGHT PEN HANDLER CONT.
33.2
33.3        J1:         MDAR        TXCNT
33.4                    MDX0        TW0
33.5                    JPLS        J2
33.6                    JPSR        LINE1
33.7                    MDAR        ONE
33.10                   ARMD        ERASEFG
33.11                   JUMP        JP
33.12
33.13       J2:         MDAR        TXCNT
33.14                   MDX0        THREE
33.15                   JPLS        J3
33.16                   JPSR        LINE1
33.17                   MDAR        ONE
33.20                   ARMD        CURFG
33.21                   ARMD        REFG
33.22                   MDAR        REFPT
33.23                   ARMD        TRCRD
33.24                   JUMP        JP
33.25
33.26       J3:         MDAR        TXCNT
33.27                   MDX0        FOUR
33.30                   JPLS        J4
33.31                   JPSR        LINE1
33.32                   MDAR        ONE
33.33                   ARMD        TRANFG
33.34                   ARMD        CURFG
33.35                   MDAR        REFPT
33.36                   ARMD        TRCRD
33.37                   JUMP        JP
33.40
33.41       J4:         MDAR        TXCNT
33.42                   MDX0        FIVE
33.43                   JPLS        J5
33.44                   JPSR        LINE1
33.45                   MDAR        ONE
33.46                   ARMD        DASHFG
33.47                   JUMP        JP
33.50
33.51       J5:         MDAR        TXCNT
33.52                   MDX0        SIX
33.53                   JPLS        J6
33.54                   JPSR        LINE1
33.55                   MDAR        ONE
33.56                   ARMD        ZOOMFG
33.57                   JUMP        JP
33.60
34.1        (TEXT LIGHT PEN HANDLER CONT.
34.2
34.3        J6:         MDAR        TXCNT
34.4                    MDX0        SEVEN
34.5                    JPLS        J7
34.6                    JPSR        LINE1
34.7                    MDAR'X      FRAMEFG
34.10                   ARX0'F
34.11                   ARMD        COUNT
34.12                   JUMP        JP
34.13
34.14       J7:         MDAR        TXCNT
34.15                   MDX0        TEN
34.16                   JPLS        J10
34.17                   JPSR        LINE1
34.20                   MDAR        ONE
34.21                   ARMD        NAMEFG
34.22                   ARX0'F
34.23                   ARMD        COUNT
34.24                   JUMP        JP
34.25
34.26       J10:        MDAR        TXCNT
34.27                   MDX0        ELEVEN
34.30                   JPLS        JP
34.31                   MDAR        ONE
34.32                   ARMD        TTYFG
34.33                   ARX0'F
34.34                   ARMD        CURFG
34.35       JP:         MDAR        LPSAV
34.36                   JUMP'I      LPLER
34.37
34.40
34.41
34.42       ( END OF LIST HANDLER
34.43
34.44       ( ROUTINE SETS END OF LIST FLAG TO A -0 SO THAT DRVEC WILL STOP -
34.45       ( LOOPING AND DRAW THE NEXT IMAGE
34.46
34.47       EOLER:      0
34.50                   ARMD        SAR1
34.51                   ARMD'0      EOLFG
34.52                   MDAR        SAR1
34.53                   JUMP'I      EOLER
34.54
```

```
35.1        [ FRAME CLOCK HANDLER
35.2
35.3        [ ROUTINE HANDLES FRAME CLOCK INTERRUPTS,IF IMAGE DONE FLAG (IDFLG) •
35.4        [ IS SET JUMP TO DTEXT AND REFRESH,IF NOT JUMP RIGHT BACK TO THE •
35.5        [ FRAMECLOCK OCCURRED
35.6
35.7        FCLER:      0
35.10                   ARMD        SAR3
35.11                   MDAR        IDFLG
35.12                   JPLS        •+3
35.13                   MDAR        SAR3
35.14                   JUMP'I      FCLER
35.15                   MDAR        LPFLG
35.16                   JPLS        •+2
35.17                   MDAR'X      LPCNT
35.20                   MDAR        FCLER
35.21                   MDAR'A      MASK5
35.22                   MDAR'B      MASK8
35.23                   ARMD        DTEXT
35.24                   MDAR'F      DTEXT+1
35.25                   ARMD        FCLER
35.26                   MDAR        SAR3
35.27                   ARMD        SAVEAR
35.30                   JUMP'I      FCLER
35.31
35.32       [ END OF TEXT STRING HANDLER
35.33
35.34       [ TXCNT•COUNTER WHICH IS INCREMENTED AFTER EVERY STRING
35.35
35.36       [ROUTINE JUSTS INCREMENTS A TEXT HANDLER SO WHEN A LIGHT PEN PICK •
35.37       [ OCCURS LPLER CAN DETERMINE WHICH FUNCTION WAS SELECTED•SENDS LCG •
35.40       [ TO NEXT STRING UNLESS FINISHED WHICH IN THAT CASE SENDS CONTROL TO •
35.41       [ DRVEC TO DRAW VECTORS•
35.42
35.43       TXLER:      0
35.44                   ARMD        SAR2
35.45                   MDAR'X      TXCNT1
35.46                   MDXB        TWELVE
35.47                   JPLS        JMP6
35.50                   MD10'A      MASK2
35.51                   MDAR        ONE
35.52                   ARMD        TXCNT1
35.53                   MDAR        TXLER
35.54                   MDAR'A      MASK5
35.55                   MDAR'B      MASK8
35.56                   ARMD        DRVEC
35.57                   MDAR'F      DRVEC+1
35.60                   ARMD        TXLER
35.61                   MDAR        SAR2
35.62                   ARMD        SAVAR
35.63                   JUMP'I      TXLER
35.64
36.1        [ TEXT HANDLER CONT•
36.2
36.3        JMP6:       MDAR        INCTXT
36.4                    MDAE        FOUR
36.5                    ARMD        INCTXT
36.6                    ARMD        77735
36.7                    MDIC'A      CM14
36.10                   MDIC'B      TEN
36.11                   MDAR        SAR2
36.12                   JUMP'I      TXLER
36.13
36.14
36.15       [ RESET LINE FLAGS ROUTINE
36.16
36.17       [ ROUTINE TO TURN OFF CURSOR AND LINE FUNCTION FLAGS
36.20
36.21       LINE1:      JUMP        •
36.22                   ARXB'F
36.23                   ARMD        LINEFG
36.24                   ARMD        CURFG
36.25                   MDIR        LINE1
```

```
37.1        [PEN TRACKING SUBROUTINE
37.2
37.3        [ PTRCP=OFFSET TABLE POINTR
37.4        [ TRCRD=CELL WHICH HOLDS CURRENT COORDS. OF CENTER OF CURSOR (TRACK COORD.)
37.5        [ TRCRP=CELL WHICH SAVES INITIAL TRCRD TO UP DATE
37.6
37.7        [ ROUTINE DRAWS A CURSOR WHICH HAS A POINT IN CENTER ENCLOSED BY A RECTANGLE -
37.10       [ WHICH IS ENCLOSED BY A DECAGON.WHEN USER HITS A SIDE OF THE RECTANGLE OR -
37.11       [ DECAGON,THE CENTER OF THE CURSOR IS MOVED THE DISTANCE AND DIRECTION OF THE -
37.12       [ OFFSET.
37.13
37.14       PTRAC:      JUMP            .
37.15                   MDAR'F          NULZ
37.16                   ARMD            E0LPV
37.17                   MDAR'L
37.20                   ARXR'F
37.21                   ARMD            LPN2
37.22                   ARX8'F
37.23                   ARMD            AR0PV
37.24                   MDAR'F          LPN?
37.25                   ARMD            LPNPV
37.26                   MD11'L
37.27                   37777JH                         [SET SCALE TO MAXIMUM
37.30                   MD10'8'L
37.31                   61420JH                         [TURN ON
37.32                   MD06'F          0
37.33                   MDAR'F          PTPT-1
37.34                   ARMD            PTRCP
37.35                   MDAR            TRCRD
37.36                   ARMD            TRCRP           [ SAVE FOR UP DATE
37.37                   MDAR'A          CM1
37.40                   MDAR'8'H        9NE
37.41                   JPSR            L0D5
37.42                   0;0;0;0;0;
37.43                   JPSR            L0D5
37.44                   MDAR            TRCRD
37.45                   MDAR'8          C1H1
37.46                   N00P
37.47                   N00P
37.50                   JPSR            L0D5
37.51                   MDAR            TRCRD
37.52                   MDAE'L
37.53                   400JH           400
37.54                   MDAR'A          CM1
37.55                   MDAR'8'H        9NE
37.56                   N00P
37.57                   N00P
37.60                   JPSR            L0D5
37.61       IREPEAT Q,(774000040l,774007740l,40077401,40000401)
37.62                   MDAR            TRCRD
37.63                   MDAE'L
37.64                   0
37.65                   MDAR'8          C1H1
40.1        [ PEN TRACK. CONT.
40.2
40.3                    N00P
40.4                    N00P
40.5                    N00P
40.6                    ARAR'X          PTRCP
40.7                    JPSR            L0D5
40.10       ENDI
40.11                   N00P
40.12                   N00P
40.13                   N00P
40.14                   N00P
40.15                   MDAR            LPN2
40.16                   JPLS            PTCK9
40.17                   MDAR'F          LPN22
40.20                   ARMD            LPNPV
40.21                   MDAR            TRCRD
40.22                   MDAE'L
40.23                   1750JH          500
40.24                   MDAR'A          CM1
40.25                   MDAR'8'H        9NE
40.26                   JPSR            L0D5
40.27       N0CARRET
40.30       IREPEAT Q,(115201523,2033,7662401523,7603000501,
40.31                  7603077301,7662476255,76745,115276255,
40.32                  175077301,1750000501);
40.33       CARRET
40.34                   N00P
40.35                   N00P
40.36                   N00P
40.37                   N00P
40.40                   MDAR            TRCRD
40.41                   MDAE'L
40.42                   0
40.43                   MDAR'8          C1H1
40.44                   ARAR'X          PTRCP
40.45                   JPSR            L0D5
40.46       ENDI
40.47                   N00P
40.50                   N00P
40.51                   N00P
40.52                   N00P
40.53       PTCK9:      MDAR            TRCRP
40.54                   ARMD            TRCRD
40.55                   MDAR'F          NULX
40.56                   ARMD            LPNPV
40.57                   MD10'A'L
40.60                   1000JH
40.61                   MDIR            PTRAC
40.62                                   0
```

```
41.1        [ PEN TRACK CONT.
41.2
41.3        PTPT:       SI;MSIJH;MSI;SIJH
41.4
41.5
41.6                    145101010;57601777;7671201777;7632701012
41.7                    76030777001;76327767761;76712760001;57676000
41.10                   145076770;175077700
41.11                   N88P;N88P;
41.12
41.13       [ LIGHT PEN HANDLERS IN PEN TRACK
41.14
41.15       [ LIGHT PEN HANDLER FOR HIT ON THE RECTANGLE
41.16
41.17       LPN2:       0
41.20                   ARMD            LPN2A
41.21                   MDAR            TRCRP
41.22                   MDAE'I          PTRCP
41.23                   ARMD            TRCRP
41.24                   MDAR            LPN2A
41.25                   JUMP'I          LPN2
41.26
41.27       [ LIGHT PEN HANDLER FOR A HIT ON THE DECAGON
41.30
41.31       LPN22:      0
41.32                   ARMD            LPN2A
41.33                   MDAR            TRCRP
41.34                   MDAE'I          PTRCP
41.35                   ARMD            TRCRP
41.36                   MDAR'F          NULX
41.37                   ARMD            LPNPV
41.40                   MDAR            LPN2A
41.41                   JUMP'I          LPN22
41.42
41.43
41.44       [ ROUTINE TO DRAW THE LINES IN THE CURSOR
41.45
41.46       LODS:       JUMP            .
41.47                   ARMD'LJ         0
41.50                   MDOS            .+1
41.51                   MDIR            LODS
41.52
41.53       NULZ:       JUMP            .
41.54                   JUMP'I          NULZ
41.55       NULX:       JUMP            .
41.56                   JUMP'I          NULX
42.1        [ DESIGNATE POINT ROUTINE
42.2
42.3
42.4        [ ROUTINE HANDLES LOADING IMAGE DIRECTORY AND WORDS (DESIGNATED BY FUNC. SWITCH1-
42.5        [ INTO THE DISPLAY LIST.IF NOT FIRST WORD IN NEW IMAGE, THE EOL BIT IS RESET IN -
42.6        [ IN THE LAST WORD OF THE DISPLAY LIST AND THE NEW WORD ADDED WITH EOL BIT SET.
42.7        [ FIRST END POINT IS A MOVE (COUNT=0),SUCCEEDING END POINTS                ARE DRAWS (COUNT -
42.10       [ GREATER THAN 01.TELETYPE INPUTS ARE SENT BY TTY TO BE ADDED TO THE DISPLAY LIST-
42.11       [ AND THE IMAGE DIRECTORY IS LOADED.
42.12
42.13       DESPT:      0
42.14                   MDAR            FLG1
42.15                   JPLS            DTEXT1
42.16                   MDAR            ONE
42.17                   ARMD            FLG1
42.20                   MDAR            LINEFG
42.21                   JPLS            JU
42.22                   MDAR            TRANFG
42.23                   JPLS            JPTR
42.24                   MDAR            REFG
42.25                   JPLS            JPREF
42.26                   JUMP            DTEXT1
42.27       JU:         MDAR            COUNT           [ THIS PORTION FOR LINE
42.30                   ARMD            TEMP1
42.31                   MDAR            DATA1
42.32                   MDX0            MZERO
42.33                   JPLS            P1
42.34                   MDAR'F          DATA1-1
42.35                   ARMD            TEMP2
42.36                   MDAR            DBLK
42.37                   ARMD            TEMP3
42.40                   JUMP            P2
42.41       P1:         ARX0'F
42.42                   ARMD            ICFG
42.43                   JPSR            WBLOP
42.44                   MDAR            WBADR
42.45                   ARMD            TEMP3
42.46                   MDAR'I          TEMP3
42.47                   MDX0            ONE
42.50                   JPLS            .+2
42.51                   JUMP            .+3
42.52                   MDAR            ONE
42.53                   ARMD            JMPFG
```

```
43.1        { DESIGNATE POINT CONT.
43.2
43.3                    MDAR'F          DATA1
43.4                    MDAE            TWCNT
43.5                    MDAE            WCONT
43.6                    MDAE'N          ONE
43.7                    ARMD            TEMP2
43.10                   MDAR            JMPFG
43.11                   JPLS            P2
43.12                   MDAR'I          TEMP2
43.13                   MDAR'A          MASK7
43.14                   ARMD'I          TEMP2
43.15
43.16       P2:         MDAR'X          TEMP3
43.17                   MDAR            SCAL
43.20                   ARMD'I          TEMP3
43.21                   MDAR'X          TEMP3
43.22                   MDAR            INTENS
43.23                   ARMD'I          TEMP3
43.24                   MDAR'X          TEMP3
43.25                   MDAR            REFPT
43.26                   ARMD'I          TEMP3
43.27                   ARX0'F
43.30                   ARMD            JMPFG
43.31                   MDAR            TEMP1
43.32                   JPLS            JU1
43.33                   MDAR            TEMP2
43.34                   MDAE            ONE
43.35                   ARMD            TEMP2
43.36                   MDAR            TEMP3
43.37                   ARMD            TEMP6
43.40                   MDAR            LFG1
43.41                   JPLS            .+3
43.42                   MDAR            TRCRD
43.43                   JUMP            P3
43.44                   MDAR            PTCORD1
43.45       P3:         MDAR'A          MASK6
43.46                   ARMD            NUMB2
43.47                   MDAR            REFPT
43.50                   ARMD            NUMB1
43.51                   JPSR            SUB2
43.52                   MDAR            NUMB1
43.53                   MDAR'0          MASK9
43.54                   ARMD'I          TEMP2
43.55                   MDAR            TEMP6
43.56                   MDAE'N          THREE
43.57                   ARMD            TEMP6
43.60                   MDAR'I          TEMP6
43.61                   MDX0            MZERO
43.62                   JPLS            .+3
43.63                   MDAR            ONE
43.64                   ARMD'I          TEMP6
44.1        { DESIGNATE POINT CONT.
44.2
44.3                    MDAR'X          COUNT
44.4                    MDAR'X          WCONT
44.5                    MDAR            LFG1
44.6                    JPLS            JU
44.7                    JUMP            DTEXT1
44.10
44.11       JU1:        MDAR            TEMP2           {THIS PORTION FOR SUCCEEDING END POINTS
44.12                   MDAE            ONE
44.13                   ARMD            TEMP2
44.14                   MDAR            LFG1
44.15                   JPLS            .+3
44.16                   MDAR            TRCRD
44.17                   JUMP            P4
44.20                   MDAR            PTCORD2
44.21       P4:         MDAR'A          MASK6
44.22                   ARMD            NUMB2
44.23                   MDAR            REFPT
44.24                   ARMD            NUMB1
44.25                   JPSR            SUB2
44.26                   MDAR            NUMB1
44.27                   MDAR'0          C1H1
44.30                   ARMD'I          TEMP2
44.31                   MDAR'X          WCONT
44.32                   MDAR            LFG1
44.33                   JPLS            T3
44.34                   JUMP            DTEXT1
44.35       JPTR:       MDAR            TRCRD           { THIS PORTION FOR TRAN
44.36                   MDAR'A          MASK6
44.37                   ARMD            NUMB2
44.40                   ARX0'F
44.41                   ARMD            TRANFG1
44.42                   ARMD            CURFG
44.43                   ARMD            TRCRD
44.44                   JUMP            DTEXT1
44.45       JPREF:      MDAR            TRCRD           { THIS PORTION FOR REF
44.46                   MDAR'A          MASK6
44.47                   ARMD            NUMB2
44.50                   ARX0'F
44.51                   ARMD            REFG1
44.52                   ARMD            CURFG
44.53                   ARMD            TRCRD
44.54                   JUMP            DTEXT1
```

```
45.1        [ WHAT BLOCK OPEN ROUTINE
45.2
45.3        [ WBAOR=ADDRESS OF HEADER OF THE OPEN IMAGE
45.4        [TBCNT=COUNT OF IMAGES IN CURRENT SUBPIC PLUS 0,10,OR 20 DEPENDING IF SUBPIC1 =
45.5        [ OR SUBPIC2 OR SUBPIC3 OPEN
45.6        [ DBLK=ADDRESS OF HEAD OF DIRECTORY
45.7
45.10       [ ROUTINE DETERMINES WHAT IMAGE (BLOCK) IS OPEN BY COMPUTING AN OFFSET FROM
45.11       [ THE HEAD OF THE DIRECTORY.
45.12
45.13       WBLOP:      JUMP        .
45.14       W1:         MOAR        TBCNT
45.15                   ARLS        2
45.16                   NOOP
45.17                   MDAE        TBCNT
45.20                   MDAE        TBCNT
45.21                   MDAE'N      SIX
45.22                   ARMD        TEMP3
45.23                   MDAR        DBLK
45.24                   MDAE        TEMP3
45.25                   ARMO        WBADR
45.26                   MDAR'I      WBAOR
45.27                   MDXO        ONE
45.30                   JPLS        .+2
45.31                   MOIR        WBLOP
45.32                   MOAR        ONE
45.33                   ARMD        ICFG
45.34                   MOAR        WBAOR
45.35                   MOAE        SIX
45.36                   ARMO        WBAOR
45.37                   MOIR        WBLOP
45.40
45.41       [CONSTANTS AND VARIABLES
45.42
45.43       TI = 300
45.44       MTI = -TI 77777
45.45       SI = 40
45.46       MSI = -SI 77777
45.47       SAR1:0
45.50       SAR2:0
45.51       SAR3:0
45.52       SAVAR:0
45.53       SAVEAR:0
46.1        [ CONSTANTS ETC. CONT.
46.2
46.3        MZERO:-0
46.4        ZERO:0
46.5        ONE:1
46.6        TWO:2
46.7        THREE:3
46.10       FOUR:4
46.11       FIVE:5
46.12       SIX:6
46.13       SEVEN:7
46.14       TEN:10
46.15       ELEVEN:11
46.16       TWELVE:12
46.17       FIFT:15
46.20       SEVTEEN:17
46.21       TWEN:20
46.22       THIR:30
46.23       TEMP1:0
46.24       TEMP2:0
46.25       TEMP3:0
46.26       TEMP4:0
46.27       TEMP5:0
46.30       TEMP6:0
46.31       C1H1:0000100001
46.32       CM1:-1
46.33       CM14:-14
46.34       INCTXT:0
46.35       TXCNT:0
46.36       TXCNT1:1
46.37       TWCNT:0
46.40       WCONT:0
46.41       BCNT:0
46.42       CNTER:0
46.43       CONT:10
46.44       CONT1:0
46.45       COUNT:0
46.46       PTCNT:0
46.47       TBCNT:0
46.50       TTYCNT:0
46.51       WBADR:0
46.52       ZCNTER:0
46.53       CURFG:0
46.54       OASHFG:0
46.55       DBLK:0
46.56       EOLFG:0
46.57       ERASEFG:0
46.60       ERRFG:0
```

```
47.1        ( CONSTANTS ETC. CONT.
47.2
47.3        FRAMEFG:0
47.4        FLG1:0
47.5        FFG1:0
47.6        FFG2:0
47.7        FFLAG:0
47.10       FRFG1:0
47.11       FRFG2:0
47.12       ICFG:0
47.13       ICFG1:0
47.14       ICFG2:0
47.15       ICFG3:0
47.16       IOFLG:0
47.17       IMC:0
47.20       IMCFG:0
47.21       JMPFG:0
47.22       LFG1:0
47.23       LINEFG:0
47.24       MOVFG:0
47.25       NAMEFG:0
47.26       LPCNT:0
47.27       LPFLG:1
47.30       LPSAV:0
47.31       LPL2A:0
47.32       PTRCR:0
47.33       NEGFG:0
47.34       RFFG:0
47.35       REFG1:1
47.36       TRANFG:0
47.37       TRANFG1:1
47.40       TTYFG:0
47.41       TYPEFG:0
47.42       ZSUMFG:0
47.43       TRCRD:0
47.44       TRCRP:0
47.45       SCAL:           37777J.
47.46       MXSCL:          37777J.
47.47       MXSC1:          37777
47.50       INTFLG:         20000
47.51       TXDY:0
47.52       NAMER:0
47.53       FAC1:0
47.54       TEMPT:0
47.55       NUMB1:0
47.56       NUMB2:0
47.57       LP2:0
47.60       PTCRRD1:0
47.61       PTCRRD2:0
50.1        MASK1:          7777777776
50.2        MASK2:          7775777777
50.3        MASK3:          0000077776
50.4        MASK4:          7777700000
50.5        MASK5:          0000077777
50.6        MASK6:          7777677776
50.7        MASK7:          7777677777
50.10       MASK8:          0004000000
50.11       MASK9:          0000100000
50.12       MASK10:         7777777776
50.13       MASK11:         0000077757
50.14       MASK12:         0377777777
50.15       MASK13:         20
50.16       MASK14:         7577777777
50.17       DASHMASK:020000/4
50.20       LFMASK:200
50.21       CLKPV = 77755; LPNPV =77760
50.22       PSVPV = 77756; EPLPV = 77757
50.23       ARGPV = 77771
```

```
51.1        [ TEXT DISPLAY LIST
51.2
51.3        ORTEXT:         0
51.4                        0470005464
51.5                        4622247212
51.6                        0000100000
51.7                        0
51.10                       0002260026
51.11                       1721251202
51.12                       5161300000
51.13                       0
51.14                       0470005504
51.15                       5121243000
51.16                       0000100000
51.17                       0
51.20                       0470005514
51.21                       5224440634
51.22                       0000100000
51.23                       0
51.24                       0470005524
51.25                       4220251620
51.26                       0000100000
51.27                       0
51.30                       0470005534
51.31                       5523647632
51.32                       2665152000
51.33                       0
51.34                       0470005544
51.35                       4324440632
51.36                       4240100000
51.37                       0
51.40                       0470005554
51.41                       4720246612
51.42                       2665152000
51.43                       0
51.44                       0470005564
51.45                       5225054400
51.46                       0000100000
51.47
51.50       [ INSTRUCTION DEFINITIONS
51.51
51.52       ^D05=25000JH
51.53       ^D06=26000JH
51.54       ^D07=27000JH
51.55       ^D10=30000JH
51.56       ^D11=31000JH
52.1        [ DIRECTORY FOR IMAGES
52.2
52.3        DBLK1:          .0
52.4                        0
52.5                        0
52.6                        0
52.7                        0
52.10                       0
52.11       DBLK2:          .0
52.12                       0
52.13                       0
52.14                       0
52.15                       0
52.16                       0
52.17       DBLK3:          .0
52.20                       0
52.21                       0
52.22                       0
52.23                       0
52.24                       0
52.25       DBLK4:          .0
52.26                       0
52.27                       0
52.30                       0
52.31                       0
52.32                       0
52.33       DBLK5:          .0
52.34                       0
52.35                       0
52.36                       0
52.37                       0
52.40                       0
52.41       DBLK6:          .0
52.42                       0
52.43                       0
52.44                       0
52.45                       0
52.46                       0
52.47       DBLK7:          .0
52.50                       0
52.51                       0
52.52                       0
52.53                       0
52.54                       0
52.55       DBLK10:         .0
52.56                       0
52.57                       0
52.60                       0
52.61                       0
52.62                       0
```

```
53.1          [ DIRECTORY FOR IMAGES IN SUBPIC2
53.2
53.3          OBLK21:     .0
53.4                      0
53.5                      0
53.6                      0
53.7                      0
53.10                     0
53.11         OBLK22:     .0
53.12                     0
53.13                     0
53.14                     0
53.15                     0
53.16                     0
53.17         OBLK23:     .0
53.20                     0
53.21                     0
53.22                     0
53.23                     0
53.24                     0
53.25         OBLK24:     .0
53.26                     0
53.27                     0
53.30                     0
53.31                     0
53.32                     0
53.33         OBLK25:     .0
53.34                     0
53.35                     0
53.36                     0
53.37                     0
53.40                     0
53.41         OBLK26:     .0
53.42                     0
53.43                     0
53.44                     0
53.45                     0
53.46                     0
53.47         OBLK27:     .0
53.50                     0
53.51                     0
53.52                     0
53.53                     0
53.54                     0
53.55         OBLK210:    .0
53.56                     0
53.57                     0
53.60                     0
53.61                     0
53.62                     0
54.1          [ DIRECTORY FOR IMAGES IN SUBPIC3
54.2
54.3          OBLK31:     .0
54.4                      0
54.5                      0
54.6                      0
54.7                      0
54.10                     0
54.11         OBLK32:     .0
54.12                     0
54.13                     0
54.14                     0
54.15                     0
54.16                     0
54.17         OBLK33:     .0
54.20                     0
54.21                     0
54.22                     0
54.23                     0
54.24                     0
54.25         OBLK34:     .0
54.26                     0
54.27                     0
54.30                     0
54.31                     0
54.32                     0
54.33         OBLK35:     .0
54.34                     0
54.35                     0
54.36                     0
54.37                     0
54.40                     0
54.41         OBLK36:     .0
54.42                     0
54.43                     0
54.44                     0
54.45                     0
54.46                     0
54.47         OBLK37:     .0
54.50                     0
54.51                     0
54.52                     0
54.53                     0
54.54                     0
54.55         OBLK310:    .0
54.56                     0
54.57                     0
54.60                     0
54.61                     0
54.62                     0
```

```
55.1        ( DIRECTORY FOR SUBPIC1
55.2
55.3        SUBP1:          0
55.4                        0
55.5                        0
55.6                        0
55.7                        0
55.10       ACNT1:          0
55.11       TBCN1:          0
55.12
55.13       ( DIRECTORY FOR SUBPIC2
55.14
55.15       SUBP2:          0
55.16                       0
55.17                       0
55.20                       0
55.21                       0
55.22       ACNT2:          0
55.23       TBCN2:          0
55.24
55.25       ( DIRECTORY FOR SUBPIC3
55.26
55.27       SUBP3:          0
55.30                       0
55.31                       0
55.32                       0
55.33                       0
55.34       ACNT3:          0
55.35       TBCN3:          0
55.36
55.37       DATA1:=0
55.40       TERMINATE
```

AGAIN1	25.21	E3	17.37	IM2	30.5	MXSCL	47.46
AGAIN2	13.30	E4	17.53	IMCL1	30.46	MZERO	46.3
AGAIN3	17.43	ELEVEN	46.15	IMCL8	27.40	N1	10.47
AGAIN	10.25	ELPLER	16.40	IMC	47.17	N2	11.3
BCNT	46.41	END	22.10	IMC10	31.22	N3	11.17
C1=1	46.31	END1	23.40	IMC1	30.60	N4	11.32
CHECK	22.20	END2	6.53	IMC2	31.12	N5	11.42
CM1	46.32	END3	17.57	IMCFG	47.20	N6	11.51
CM14	46.33	ERLER	34.47	INCTXT	46.34	NAME1	10.14
CNTER	46.42	ERLFG	46.56	INTENS	47.50	NAMEFG	47.25
COMP1	14.56	ERASE1	16.6	J10	34.26	NAMER	47.52
COMP	14.43	ERASEFG	46.57	J1	33.3	NEGFG	47.33
CONT1	46.44	ERROR	26.47	J2	33.13	NULX	41.55
CONT	46.43	ERR1	12.3	J3	33.26	NULZ	41.53
COUNT	46.45	ERR2	15.40	J4	33.41	NUMB1	47.55
CURFG	46.53	ERR3	15.53	J5	33.51	NUMB2	47.56
D1	27.13	ERR4	15.61	J6	34.3	ONE	46.5
DASH1	27.7	ERRFG	46.60	J7	34.14	OVER1	11.46
DASHFG	46.54	F1	5.35	JMP6	36.3	OVER	5.51
DASHMASK	50.17	F2	6.16	JMPFG	47.21	P1	42.41
DATA1	55.37	FAC1	47.53	JPREF	44.45	P2	43.16
DBLK10	52.55	FCLER	35.7	JP	34.35	P3	43.45
DBLK1	52.3	FFG1	47.5	JPTR	44.35	P4	44.21
DBLK210	53.55	FFG2	47.6	JU1	44.11	PTCK9	40.53
DBLK21	53.3	FFLAG	47.7	JU	42.27	PTCNT	46.46
DBLK22	53.11	FIFT	46.17	LFG1	47.22	PTCORD1	47.60
DBLK23	53.17	FIN11	15.3	LINE1	36.21	PTCORD2	47.61
DBLK24	53.25	FIN12	16.23	LINEFG	47.23	PTPT	41.3
DBLK25	53.33	FIN13	27.22	LOD5	41.46	PTRAC	37.14
DBLK26	53.41	FINIS	12.7	LP2	47.57	PTRCP	47.32
DBLK27	53.47	FIVE	46.11	LPCNT	47.26	R0	21.12
DBLK2	52.11	FLG1	47.4	LPFLG	47.27	R1	21.15
DBLK310	54.55	FOUR	46.10	LPLER	32.22	R2	21.50
DBLK31	54.3	FR1	6.3	LPMASK	50.20	R3	22.3
DBLK32	54.11	FR2	6.35	LPN22	41.31	RCOMP1	20.22
DBLK33	54.17	FR5	7.24	LPN2A	47.31	RCOMP	15.16
DBLK34	54.25	FR6	7.35	LPN2	41.16	RCOM	15.13
DBLK35	54.33	FRAM1	5.17	LPSAV	47.30	REFPT	47.54
DBLK36	54.41	FRAM3	7.7	MASK10	50.12	REF1	20.17
DBLK37	54.47	FRAMEFG	47.3	MASK11	50.13	REFG1	47.35
DBLK3	52.17	FRFG1	47.10	MASK12	50.14	REFG	47.34
DBLK4	52.25	FRFG2	47.11	MASK13	50.15	SAR1	45.47
DBLK5	52.33	GPGSY	1.17	MASK14	50.16	SAR2	45.50
DBLK6	52.41	ICFG1	47.13	MASK1	50.1	SAR3	45.51
DBLK7	52.47	ICFG2	47.14	MASK2	50.2	SAVAR	45.52
DBLK	46.55	ICFG3	47.15	MASK3	50.3	SAVEAR	45.53
DESPT	42.13	ICFG	47.12	MASK4	50.4	SCAL	47.45
DPTEXT	51.3	ICFLG	47.16	MASK5	50.5	SEVTEEN	46.20
DRVEC	3.20	IM1	27.55	MASK6	50.6	SEVEN	46.13
DTEXT1	2.13	IM10	30.12	MASK7	50.7	SIX	46.12
DTEXT	1.53	IM11	30.20	MASK8	50.10	SKIP1	2.32
DXDY	47.51	IM12	30.24	MASK9	50.11	SKIP22	14.3
E1	16.46	IM13	30.30	MOVFG	47.24	SKIP23	14.27

SKIP	2.30	Z2	26.12
SUBP1	55.3	Z3	26.31
SUBP2	55.15	Z4	26.37
SUBP3	55.27	Z5	26.41
SUB2	24.11	ZCNTER	46.52
T1	14.25	ZERO	46.4
T2	14.41	ZOOM1	25.14
T3	14.46	ZOOM2	26.3
TBCN1	55.11	ZOOMFG	47.42
TBCN2	55.23		
TBCN3	55.35		
TBCNT	46.47		
TCOMP	15.31		
TEMP1	46.23		
TEMP2	46.24		
TEMP3	46.25		
TEMP4	46.26		
TEMP5	46.27		
TEMP6	46.30		
TEN	46.14		
THIR	46.22		
THREE	46.7		
TRAN1	23.14		
TRAN2	23.17		
TRAN3	23.27		
TRANFG1	47.37		
TRANFG	47.36		
TRATY	15.25		
TRCRD	47.43		
TRCRP	47.44		
TT	4.3		
TTY1	13.17		
TTYCNT	46.50		
TTYFG	47.40		
TWCNT	46.37		
TWELVE	46.16		
TWEN	46.21		
TWO	46.6		
TXCNT1	46.36		
TXCNT	46.35		
TXLER	35.43		
TYPEFG	47.41		
V1	4.34		
V2	4.36		
V3	4.43		
W1	45.14		
WAIT1	26.61		
WBADR	46.51		
WBLOP	45.13		
WCNT1	55.10		
WCNT2	55.22		
WCNT3	55.34		
WCONT	46.40		

LIST OF REFERENCES

1. Siders, R. A., and others, Computer Graphics, A Revolution in Design, American Management Association, 1966.

2. Brown, S. A., and others, "A Description of the APT Language," Communications of the ACM, v. 6, pp. 649-658, November, 1963.

3. Hurwitz, A., and others, "GRAF: Graphic Additions to FORTRAN," in AFIPS Conference Proceedings 1967 Spring Joint Computer Conference, v. 30. Thompson Book Company, 1967.

4. Thornhill, D. E., and others, An Integrated Hardware-Software System for Computer Graphics in Time-Sharing, Massachusetts Institute of Technology, November, 1968.

5. Anderson, R. H., and Farber, D. J., Extensions to the PL/I Language for Interactive Computer Graphics, The Rand Corporation, RM-6028-ARPA, January, 1970.

6. Rully, A. D., "A Subroutine Package for FORTRAN," Interactive Graphics in Data Processing, IBM Systems Journal, v. 7, pp. 248-256, 1968.

7. Gagliano, F. W., and others, "A Conversational Display Capability," Interactive Graphics in Data Processing, IBM Systems Journal, v. 7, pp. 281-291, 1968.

8. Brown, G. D., and Bush, C. H., The Integrated Graphics System for the IBM 2250, The Rand Corporation, 1968.

9. Streit, E., "VIP: A Conversational System for Computer-Aided Graphics," in Pertinent Concepts in Computer Graphics, Faiman, M., and Nievergelt, J., ed., University of Illinois Press, 1969.

10. Sutherland, I. E., "Sketchpad A Man-Machine Graphical Communication System," AFIPS Conference proceedings 1963 Spring Joint Computer Conference, v. 23, Spartan Books Incorporated, 1963.

11. Electronic Systems Laboratory, Massachusetts Institute of Technology, ESL-TM-220, Some Experiments with an Algorithmic Graphical Language, by Lang, C. A., and others, August, 1965.

12. Yarbrough, L. D., "CAFE: A nonprocedural Language for Computer Animation" in Pertinent Concepts in Computer Graphics, Faiman, M., and Nievergelt, J., ed., University of Illinois Press, 1969.

13. Ledley, R. S., and others, "BUGSYS: A Programming System for Picture Processing - Not for Debugging," Communications of the ACM, v. 9, pp. 79-84, February, 1966.

14. Miller, W. F., and Shaw, A. C., "A Picture Calculus," in Emerging Concepts in Computer Graphics, Secrest, D., and Nievergelt, J., ed., W. A. Benjamin Incorporated, 1968.

15. Kulsrud, H. E., "A General Purpose Graphic Language," Communications of the ACM, v. 11, pp. 247-254, April, 1968.

16. Morrison, R. A., "Graphic Language Translation with a Language Independent Processor," in AFIPS Conference Proceedings 1967 Fall Joint Computer Conference, v. 31, Thompson Book Company, 1967.

17. Roberts, L. G., "A Graphical Service System with Variable Syntax, Communications of the ACM, v. 9, pp. 173-175, March, 1966.

18. Coons, S. A., "An Outline of the Requirements for a Computer-Aided Design System," in AFIPS Conference Proceedings 1963 Spring Joint Computer Conference, v. 23, Spartan Books Incorporated, 1963.

19. Notely, M. G., "A Graphical Picture Drawing Language," The Computer Bulletin, v. 14, pp. 68-74, March, 1970.

20. Chen, F. C., and Dougherty, R. L., "A System for Implementing Interactive Applications," Interactive Graphics in Data Processing, IBM Systems Journal, v. 7, pp. 257-270, 1968.

21. Herzog, B., "Computer Graphics for Designers," in Emerging Concepts in Computer Graphics, Secrest, D., and Nievergelt, J., ed., W. A. Benjamin Incorporated, 1968.

22. Johnson, T. E., "Sketchpad III A Computer Program for Drawing in Three Dimensions," in AFIPS Conference Proceedings 1963 Spring Joint Computer Conference, v. 23, Spartan Books Incorporated, 1963.

23. Lincoln Laboratory, Massachusetts Institute of Technology, Technical Report Number 315, Machine Perception of Three-Dimensional Solids, by L. G. Roberts, 22 May 1963.

24. International Business Machines Corporation, IBM SYSTEM/360 Operating System Graphic Subroutine Package (GSP) for FORTRAN IV, COBOL, and PL/I, 1969.

25. Sutherland, I. E., "Computer Displays," Scientific American, v. 222, pp. 57-81, June, 1970.

26. Spiegel, M. R., Theory and Problems of Vector Analysis and an Introduction to Tensor Analysis, Schaum Publishing Company, 1959.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	LT Ronald D. DeLaura, USNR (Code 52Dl) Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Dr. George Rahe (Code 52 Ra) Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	MAJOR James Dale Beans, USMC 117 Spa View Avenue Annapolis, Maryland 21401	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3 REPORT TITLE

GPGL: A Model Interactive, General Purpose Graphic Language

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Master's Thesis; December 1971

5. AUTHOR(S) (First name, middle initial, last name)

James Dale Beans

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1971	117	26

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10 DISTRIBUTION STATEMENT

This document has been approved for public release;distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

General Purpose Graphic Language (GPGL) is an interactive language which is intended for both two-dimensional and three-dimensional displays. The thesis contains a survey of the attributes and capabilities of an interactive general purpose graphic language. The more popular general purpose graphic languages are compared and the results included. The system and user-defined functions (including the construction of user-defined functions) of GPGL are explained. The implementation of a subset of GPGL at the Naval Postgraduate School on an Adage AGT-10 graphics terminal is described. The main purpose of implementing a selected subset of functions from GPGL is to examine the tri-level hierarchy established within the components of the graphical display; the manner in which this hierarchy is implemented is addressed in the thesis.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer graphics						
Graphics language						
Interactive graphics						
Graphical language						
Interaction						
Graphics